CADEL

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

(...)

# 25 CORSO PRATICO COL COMPUTER

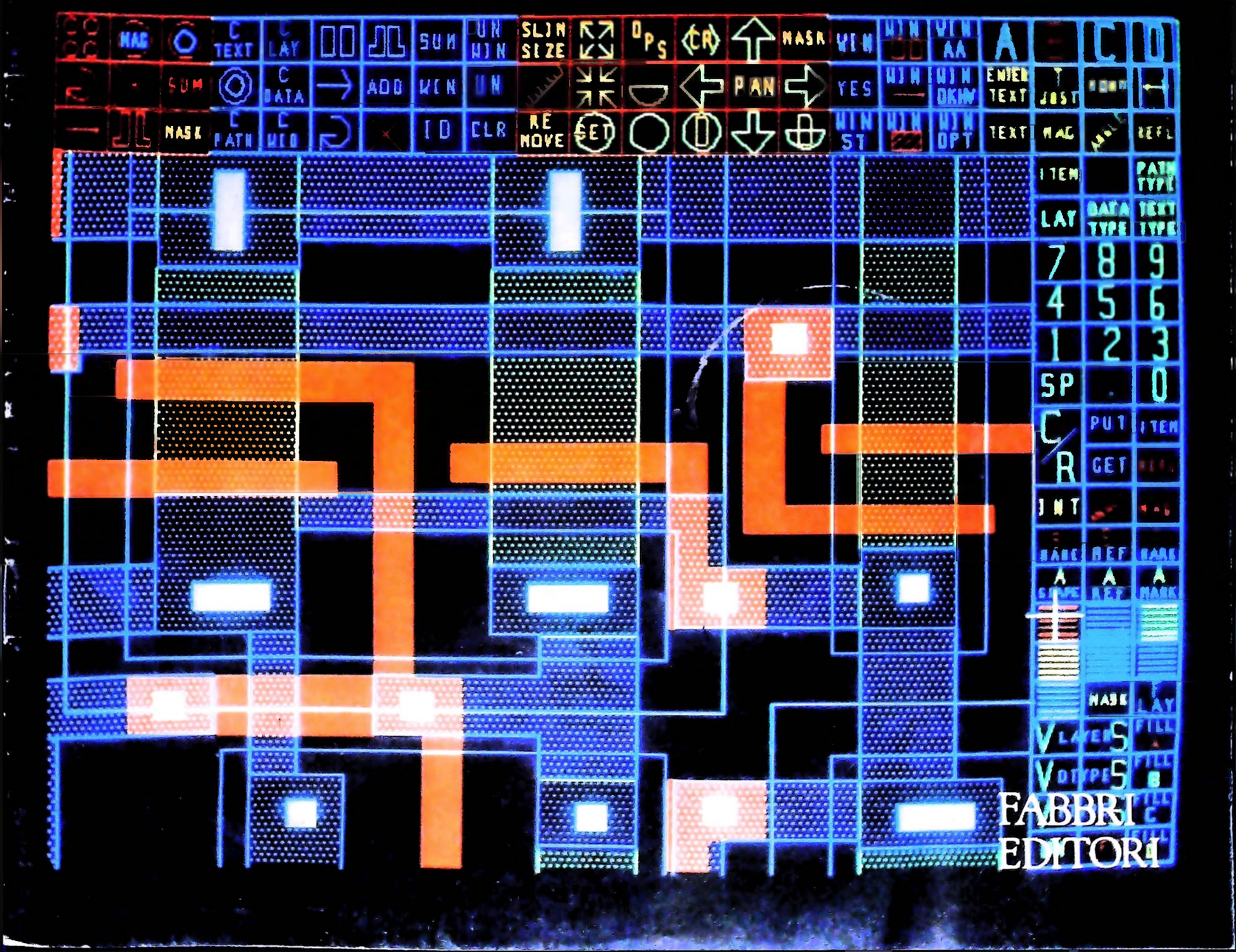
421776

è una iniziativa
**FABBRI EDITORI**

in collaborazione con
**BANCO DI ROMA**

e **OLIVETTI**

diretto da GIANNI DEGLI ANTONI



# Nel Pocket Computer PC-1500A c'è tutta la potenza di un Home Computer

PC-1500 A
dimensioni: 195 x 25,5 x 86 mm.
429.000 + IVA


INTERNORD

## Il Pocket Computer con 8,5 kbyte di RAM espandibili e una serie di interessanti accessori. Per il commerciale, il manager, l'ingegnere e l'hobbista.

### CPU - C-MOS da 8 bit

Permette un'alta velocità d'elaborazione con poco consumo d'energia.

### Mini-visore grafico

È in grado di visualizzare fino a 26 caratteri o qualsiasi tipo di rappresentazione grafica grazie ai suoi 1092 punti.

### Superiorità operativa

La **tastiera tipo macchina da scrivere** ne facilita l'uso. Inserito nell'interfaccia opzionale CE-150, il PC-1500A può essere addirittura utilizzato come una piccola macchina da scrivere!
**Sei tasti software** possono essere utilizzati come tasti funzione, o comandi, o come tasti per la definizione di programmi.
Il sistema di rotazione a tre livelli fa svolgere con 6 tasti il lavoro di 18 tasti software. La **funzione di blocco del programma** blocca il tasto MODE; il PC-1500A conserva così il programma lasciando libera solo la funzione RUN. Effettuerete così il programma senza rischiare cancellazioni involontarie.

### Linguaggio BASIC più ricco

Gli **statement aggiuntivi** del BASIC del PC-1500A forniscono variabili che possono essere liberamente definite usando uno o due caratteri, disposte secondo schemi a due dimensioni per il calcolo matriciale, stringhe di variabili, comandi per la grafica.

### Funzione orologio

Indica mese, data, ora, minuti e secondi. Ha anche una funzione acustica.
Alle scadenze programmate il computer emette dei BIP sonori e visualizza il messaggio.

### Altre caratteristiche

- Si può regolare il tono ed il numero delle ripetizioni dei BIP.
- Si possono programmare ed effettuare dei giochi.
- Abbreviazioni e dieci comandi diretti rendono la programmazione più semplice.
- Il dispositivo di spegnimento automatico evita sprechi di energia.
- Premendo i tasti SHIFT o SML si introducono lettere minuscole.
- I programmi e gli accessori per il PC-1500 sono compatibili col PC-1500A e viceversa

### Accessori opzionali

**L'interfaccia registratore/stampante grafica CE-150**

- Come stampante realizza l'hardcopy di programmi, dati e calcoli e offre una funzione grafica a quattro colori: rosso, nero, verde, blu. Stampa i caratteri in nove corpi diversi. Ogni riga ne può contenere da 4 a 36. Questa stampante può essere controllata dall'operatore orientando il senso di stampa verso l'alto, il basso, a destra e a sinistra.
- Come interfaccia si può collegare ad uno o due registratori (uno per la memorizzazione dei dati e programmi, l'altro per richiamarli)

**CE-151 - CE-155 (8 e 16 kbyte)**
**Moduli d'espansione memoria RAM**
La RAM può essere ampliata di 4 o 8 kbyte inserendo uno dei due moduli opzionali, ottenendo una capacità totale di 12,5 o 16,5 kbyte.

**CE-159 - CE-161 (8 e 16 kbyte)**
**Moduli di RAM programmabile**
I moduli CE-159 e CE-161 possono essere programmati e quindi essere rimossi.
La memoria è mantenuta per due anni se il modulo è staccato e per 5 se mantenuto nel PC-1500A.
La capacità totale, con uno di questi moduli, sarà di 16,5 o 24,5 kbyte.

**RS-232C ed interfaccia parallela CE-158**
Questa interfaccia permette la connessione del PC-1500 A con MODEM, accoppiatori acustici, lettori di codici a barre, stampanti seriali o parallele, o con altri computer.

**CE-152 Registratore a cassette**
Il CE-152 registra programmi o dati.
Si possono utilizzare due CE-152 contemporaneamente (uno per leggere, l'altro per registrare).

**CE-153 Tavoletta software**
La CE-153 ha 140 tasti definibili per facilitare l'elaborazione dei dati.

concessionaria per l'Italia
MELCHIONI

TUTTA LA POTENZA DI UN COMPUTER NEL PALMO DELLA TUA MANO

Per ulteriori informazioni scrivete a
MELCHIONI - Divisione Pocket Computer - 20138 MILANO - Via P. Colletta 37

SHARP


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

Testi
GOFFREDO HAUS, GIANCARLO MAURI,
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. - 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

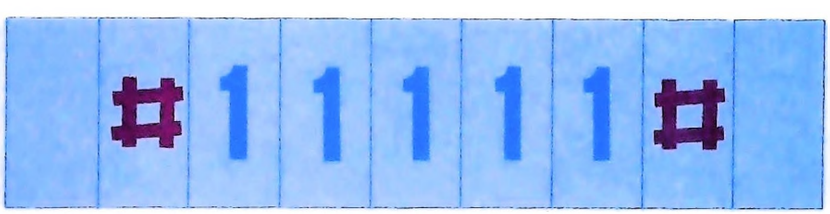
# LA MACCHINA DI TURING UNIVERSALE

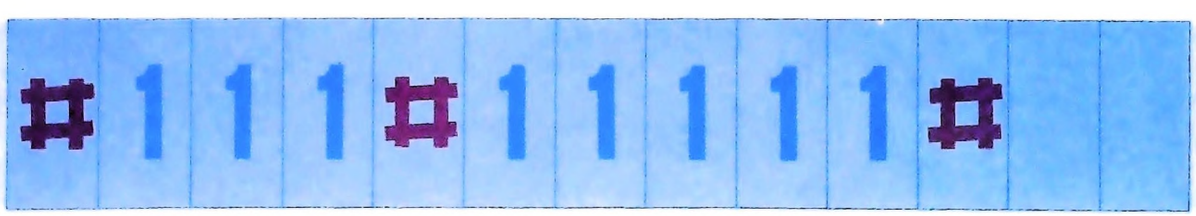
## Un "calcolatore" a programma memorizzato anticipa di dieci anni i calcolatori elettronici.

### Come funziona una macchina di Turing?

Dopo aver descritto la struttura delle macchine di Turing, descriviamone il funzionamento. È ancora Turing che ci spiega: "Noi conosciamo lo stato della computazione se conosciamo la sequenza dei simboli sul nastro, il simbolo osservato dal calcolatore ed il suo stato mentale, e questi determinano il comportamento del calcolatore ad ogni istante".
Ad ogni istante, la macchina si trova quindi in uno stato globale, o *configurazione*, che è caratterizzato da tre elementi: la sequenza $w$ di simboli scritti nella parte non bianca del nastro, lo stato $q$ del controllo e il simbolo $s$ contenuto nella cella su cui è posizionata la testina. Una configurazione è in sostanza una "fotografia" di una macchina di Turing in un certo istante del suo funzionamento. In forma sintetica, indicheremo una configurazione con $xqsy$, ove $x$ e $y$ sono le sequenze di simboli scritti rispettivamente nelle celle a sinistra e a destra della testina, $s$ è il simbolo letto e $q$ lo stato del controllo.

Sulla base dei due ultimi elementi (stato corrente e simbolo letto), la macchina determina in modo univoco, ad esempio consultando una tabella detta *tabella di transizione*, un nuovo simbolo $s'$ da sostituire ad $s$, un nuovo stato $q'$ in cui passare e la direzione in cui spostare la testina ($D$, destra, o $S$, sinistra), e provvede quindi a effettuare queste modifiche, cioè a *compiere una mossa*.
La tabella di transizione è evidentemente il principale elemento che caratterizza una macchina di Turing, poiché contiene, in forma codificata, le regole di funzionamento, il *programma* della macchina stessa. Essa rappresenta una funzione $\delta$ che ad ogni coppia (stato, simbolo) associa una tripla (nuovo stato, nuovo simbolo, direzione dello spostamento). Sulla base del contenuto della tabella, la macchina stabilisce la mossa da eseguire a ogni istante.
Formalmente, le singole mosse possono essere descritte attraverso una funzione, che indicheremo con il simbolo $\vdash$, che ad una configurazione associa una nuova configurazione in base alle seguenti regole:




Nella macchina di Turing un numero viene scritto come semplice successione di segni unari, che esprimono la "quantità". Ad esempio, il numero 5 è rappresentato dalla sequenza 1 1 1 1 1. Il segno ♯ a sinistra e a destra serve a indicare dove inizia e termina ogni successione numerica.

se $(q, s) = (q', s', \text{destra})$, allora $vcqsw \vdash vcs'q'w$
(in particolare, $vcqa \vdash vcs'q'b$)
se $(q, s) = (q', s', \text{sinistra})$, allora $vcqsw \vdash vq'cs'w$
(in particolare, $cqsw \vdash q'cs'w$)

(i simboli $s$, $s'$, $c$ rappresentano elementi di $\Sigma$, i simboli $v$ e $w$ sequenze di elementi di $\Sigma$, o parole su $\Sigma$).

Una *computazione* è una successione di mosse che portano da una data configurazione ad un'altra, e verrà indicata con il simbolo $\vdash^*$. Perché la macchina computi una funzione dobbiamo però ancora stabilire il punto di partenza e il punto di arrivo della computazione.

Come punto di partenza, si assume che il nastro contenga una sequenza finita $w$ di simboli non bianchi, che codifica i dati di ingresso in forma unaria. Ciò significa che il numero $N$ è rappresentato sul nastro da una successione di tanti uni quanto è il valore di $N$, delimitata a destra e sinistra dal simbolo ♯ (figura della pagina precedente). Si assume inoltre che la testina sia posizionata sulla cella non vuota più a sinistra e che il controllo si trovi in un particolare *stato iniziale* $q_0$: una configurazione iniziale ha quindi la forma $q_0w$. Infine, la macchina si arresta se (e quando) raggiunge uno speciale *stato finale* $q_F$, in corrispondenza del quale non è definita alcuna mossa; la sequenza di simboli scritta sul nastro al momento dell'arresto rappresenta il risultato della computazione.

Possiamo finalmente dare la definizione che più ci interessa: si dice funzione computata da una macchina di Turing M la funzione $f_M$ definita da:

$$f_M(w) = \begin{cases} z \text{ se } q_0w \vdash^* tq_Fv \text{ e } tv = z \\ \text{indefinita se M non si arresta} \end{cases}$$

(Si osservi che $f_M$ è una funzione parziale, cioè può non essere definita per alcuni valori della variabile).

Abbiamo così ottenuto quello che volevamo, cioè una definizione formale della computabilità:
una funzione f: N → N è computabile se esiste qualche macchina di Turing che la computa.

## Una macchina di Turing per la sottrazione

Come esempio, costruiamo ora una macchina di Turing che risolva il seguente problema: dati due numeri naturali $m$ ed $n$, calcolarne la differenza $m$-$n$.

Poiché per rappresentare i numeri $n$ ed $m$ usiamo la notazione unaria, col simbolo ♯ per separare i due numeri, la computazione partirà con la seguente parte di nastro non bianca:

$$\not b \quad \underbrace{1\,1\,\ldots\,1}_{m \text{ caselle}} \quad \sharp \quad \underbrace{1\,\ldots\,1\,1}_{n \text{ caselle}} \quad \not b$$

con la testina posizionata sulla casella non bianca più a sinistra e il controllo nello stato $q_0$. Al termine della computazione vogliamo avere sul nastro $m$-$n$ simboli 1 consecutivi, seguiti da ♮, mentre il resto del nastro dovrà essere bianco.

Per ottenere questo risultato, possiamo ad esempio fare in modo che la macchina cancelli alternativamente un 1 a destra (ad esempio, il più a destra) e un 1 a sinistra (ad esempio, il più a sinistra) del simbolo ♮, fino ad esaurire tutti gli 1 a destra di ♮. Se invece si esauriscono prima gli 1 a sinistra di ♮, ciò significa che $m$ è più piccolo di $n$, per cui la sottrazione non si può eseguire (si ricordi che non abbiamo i numeri negativi!): in questo caso, la macchina non si arresta.

Traduciamo questa idea in passi di computazione precisi.

1) Partendo dalla configurazione $q_0$n♯m, la macchina deve raggiungere l'ultimo simbolo a destra, per cancellarlo, attraversando tutte le altre caselle senza modificarle. Porremo allora che, se la macchina si trova in $q_0$ e legge 1 o ♯, lascia invariati sia il simbolo che lo stato e si sposta a destra.

2) La macchina può riconoscere il simbolo più a destra solo dal fatto che questo è seguito da una cella vuota; deve quindi superarlo e poi tornare indietro, cioè se nello stato $q_0$ trova $\not b$, passa in un nuovo stato $q_1$ e inizia a spostarsi a sinistra.

3) A questo punto, la macchina si trova nello stato $q_1$ e può leggere 1 o ♯. Nel primo caso cancella 1, passa in un terzo stato $q_2$ e si sposta a sinistra. Nel secondo caso, non c'è più niente da sottrarre: la sottrazione è completata, quindi la macchina passa nello stato finale e si arresta.

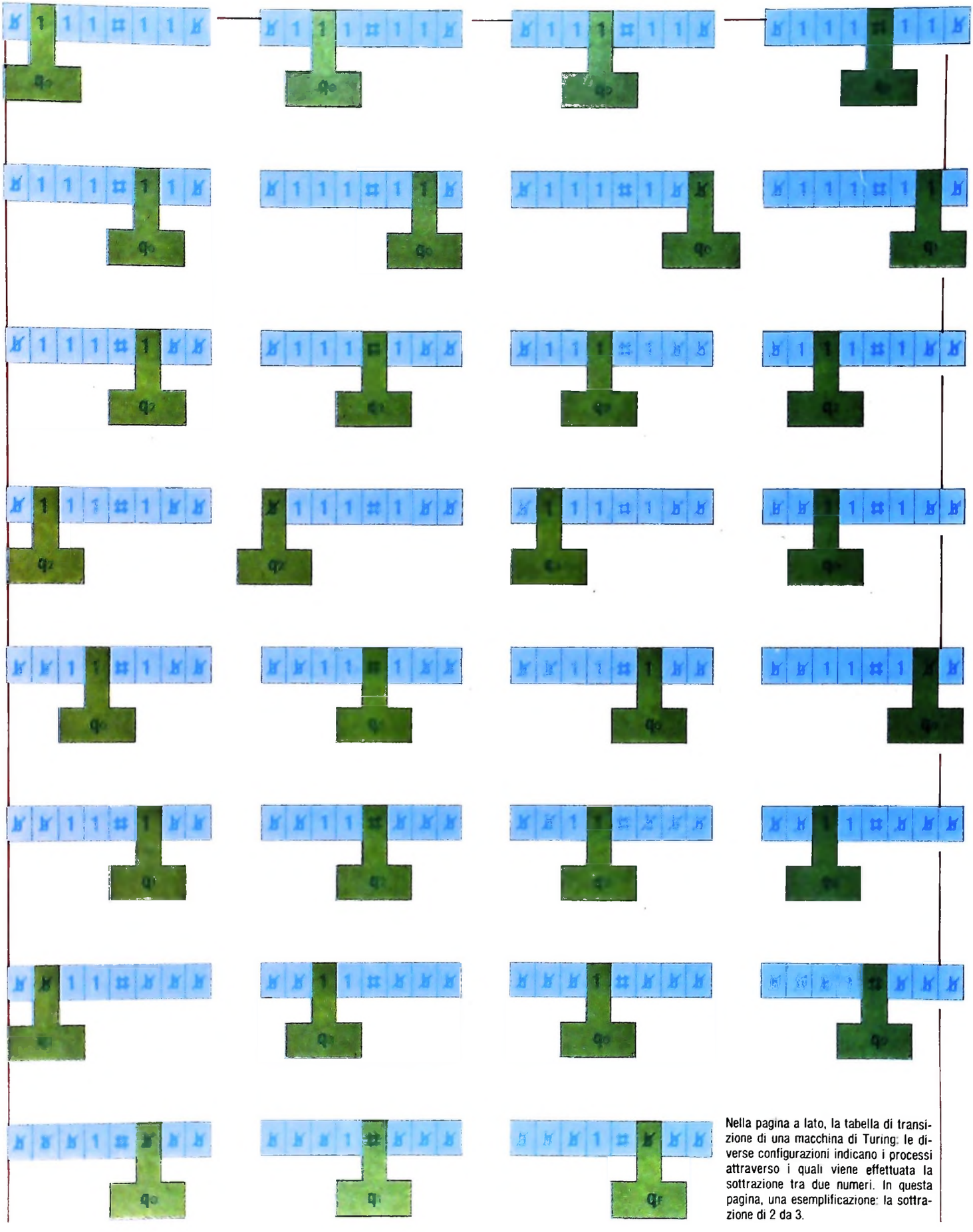
4) Nello stato $q_2$, la macchina si sposta a sinistra senza alterare né lo stato né il simbolo letto finché legge 1 o ♯; in questo modo, si riporta all'estremità sinistra del nastro scritto.

5) Come in 2), la macchina inverte la direzione quando trova la prima cella vuota. Formalmente, la lettura di $b$ nello stato $q$ provoca il passaggio in $q_3$ e lo spostamento a destra.

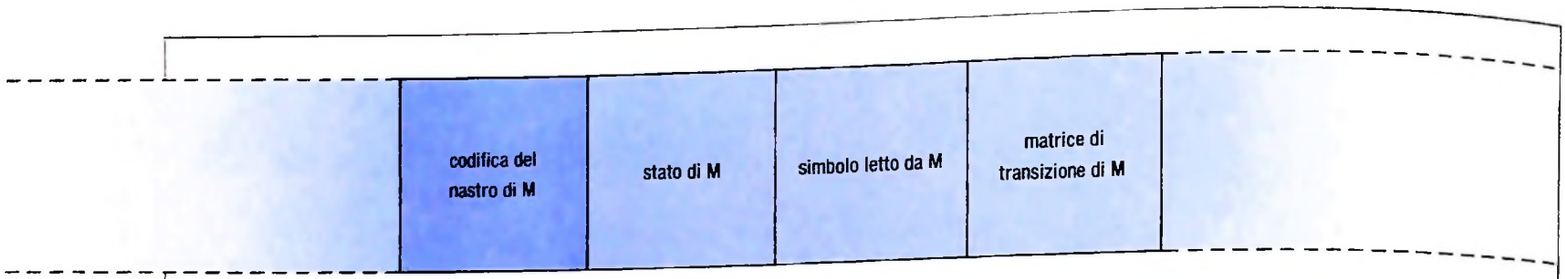
6) In $q_3$ si può leggere 1 o ♯. Nel primo caso, si sottrae 1 cancellandolo, si passa nello stato $q_0$ e si riparte da 1), per un

**TABELLA DI TRANSIZIONE**

| Configurazione corrente | | Configurazione successiva | | |
|---|---|---|---|---|
| stato | simbolo | stato | simbolo | spostamento |
| $q_0$ | 1 | $q_0$ | 1 | destra |
| $q_0$ | # | $q_0$ | # | destra |
| $q_0$ | $\not b$ | $q_1$ | $\not b$ | sinistra |
| $q_1$ | 1 | $q_2$ | $\not b$ | sinistra |
| $q_1$ | # | $q_F$ | # | destra |
| $q_1$ | $\not b$ | $q_4$ | $\not b$ | sinistra |
| $q_2$ | 1 | $q_2$ | 1 | sinistra |
| $q_2$ | # | $q_2$ | # | sinistra |
| $q_2$ | $\not b$ | $q_3$ | $\not b$ | destra |
| $q_3$ | 1 | $q_0$ | $\not b$ | destra |
| $q_3$ | # | $q_4$ | # | sinistra |
| $q_3$ | $\not b$ | $q_4$ | $\not b$ | sinistra |
| $q_4$ | 1 | $q_4$ | 1 | sinistra |
| $q_4$ | # | $q_4$ | # | sinistra |
| $q_4$ | $\not b$ | $q_4$ | $\not b$ | sinistra |



Nella pagina a lato, la tabella di transizione di una macchina di Turing: le diverse configurazioni indicano i processi attraverso i quali viene effettuata la sottrazione tra due numeri. In questa pagina, una esemplificazione: la sottrazione di 2 da 3.




Schematizzazione di un nastro della macchina di Turing universale. La parte più a destra del nastro contiene una opportuna codifica della matrice di transizione di una qualsiasi macchina *M*, che la macchina universale sta simulando, cioè il programma che $M_U$ deve eseguire. Nella parte più a sinistra del nastro è compresa una rappresentazione del contenuto del nastro di *M*. Due zone del nastro infine sono riservate agli altri due elementi della descrizione istantanea (simulata) di *M*, e precisamente lo stato di controllo e la posizione della testina.

nuovo ciclo di cancellazione. Nel secondo caso, le unità del minuendo ($m$) si sono esaurite prima delle unità del sottraendo ($n$), e la sottrazione non può essere eseguita. Si passa allora in uno speciale stato trappola $q_4$ in cui la testina continua a spostarsi ad esempio a sinistra senza alterare né lo stato né il contenuto delle celle (tabella di pag. 386).
Come si vede, per un'operazione semplice siamo stati costretti a inventare un marchingegno non certo semplice; è chiaro quindi che nessuno userà una macchina di Turing per risolvere un problema reale. Tuttavia, per affrontare problemi teorici in cui occorre un modello di riferimento rigorosissimo e uniforme, essa resta un riferimento obbligato.

## La macchina di Turing universale

Una macchina di Turing, così come l'abbiamo definita, è in grado di calcolare una sola funzione; la tabella di transizione, cioè il programma della macchina, è parte integrante della macchina stessa, e non può essere modificata senza modificare l'intera macchina (ad esempio, il numero degli stati).
Ben diverso è il comportamento dell'uomo che calcola, o degli elaboratori programmabili: essi infatti non eseguono un programma fisso, interno, ma sono in grado di eseguire qualunque programma venga loro fornito dall'esterno, purché basato su operazioni che conoscono. Ad esempio, data una macchina di Turing qualsiasi, chiunque abbia ben compreso il significato della tabella di transizione è in grado di imitarne il funzionamento a partire da un dato iniziale assegnato. Da questo punto di vista, le macchine di Turing sembrano molto più simili a calcolatrici tascabili molto specializzate che ad un calcolatore programmabile.
In realtà non è così: l'insieme delle macchine di Turing è così vario che contiene anche macchine, dette macchine di Turing universali, che sono in grado, purché si forniscano loro come dati tutte le informazioni necessarie, di imitare o simulare il comportamento di qualunque altra macchina di Turing.
Più precisamente, chiameremo macchina di Turing universale una macchina di Turing U che, dopo aver ricevuto in ingresso un'opportuna codifica della tabella di transizione di una qualsiasi altra macchina M ed una parola $w$ fornisce in uscita il valore $f_M(w)$, ove $f_M$ è la funzione computata della macchina M.
Resta da decidere come codificare le tabelle di transizione; di nuovo l'idea è quella di Gödel: a ogni macchina di Turing si associa un numero naturale che la identifica in modo univoco. È un po' come se preparassimo un catalogo delle macchine di Turing e fornissimo quindi alla macchina universale il numero di catalogo di una particolare macchina, oltre a un dato di ingresso. La macchina universale legge il numero, cerca nel catalogo le istruzioni di funzionamento della macchina M e la imita.
La capacità essenziale di una macchina universale U deve essere quella di interpretare correttamente le istruzioni di ogni altra macchina M, ricavandole dal numero di catalogo di M che U memorizza in una particolare zona del proprio nastro. Una seconda porzione di nastro è invece riservata alla simulazione vera e propria di M.
In questa parte del nastro vengono codificate le successive configurazioni istantanee assunte da M nel corso della computazione, e cioè lo stato corrente, il simbolo letto, il contenuto del nastro di M e la posizione della testina, indicata da una apposita marca. Una mossa di M si traduce in un insieme di mosse di U, che, per "imitare" il comportamento di M, consulta alternativamente la tabella di transizione e la codifica del nastro di M. Così U riesce a trovare la mossa da eseguire, e in base a questa, trasforma la codifica di una configurazione di M in quella della configurazione successiva.
Non è difficile vedere in questo comportamento strettissime analogie con i calcolatori reali. Abbiamo un programma, anche se rappresentato in forma piuttosto inconsueta per chi sia abituato ai normali linguaggi di programmazione: infatti il programma è dato in questo caso da un numero. Abbiamo poi una memoria, e infine una unità di controllo che ad ogni passo reperisce l'istruzione da eseguire, decodificando il programma, e quindi effettua le modifiche della memoria corrispondenti. Possiamo così concludere che qualunque elaboratore reale può essere simulato da una macchina di Turing universale, e che una macchina di Turing universale potrebbe in teoria essere simulata da qualunque elaboratore. In pratica, il discorso è diverso: i vincoli di memoria dei calcolatori reali (si ricordi che la memoria di una macchina di Turing è virtualmente illimitata) limitano fortemente questa possibilità.

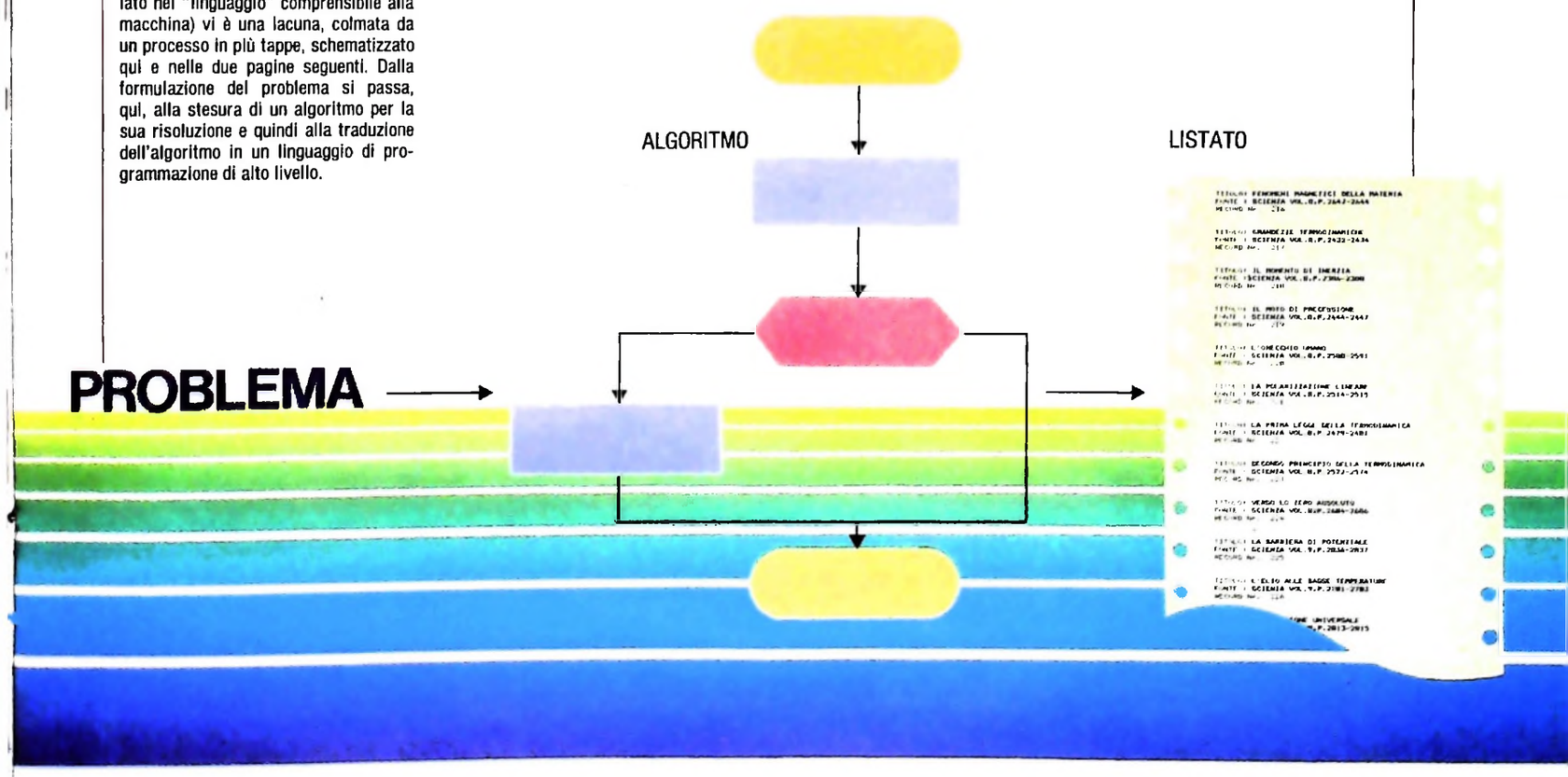
# I LINGUAGGI DI PROGRAMMAZIONE

## Sono nati per facilitare la costruzione dei programmi necessari per il calcolatore.

Da quando sono apparsi i calcolatori elettronici è sorto il problema di come comunicare a questi gli algoritmi per risolvere determinati problemi, sotto la forma di programmi.
Infatti, mentre per l'uomo è molto spontaneo esprimere un algoritmo che risolva un determinato problema in un linguaggio vicino a quello naturale (come per esempio l'italiano), facendo riferimento a una terminologia tipica del problema da risolvere, il calcolatore è un'apparecchiatura elettronica che è in grado di eseguire istruzioni molto semplici e molto lontane da quelle usate nella comunicazione umana.

Se pensiamo, per esempio, a un semplice problema di fatturazione, i termini più spontanei che un ragioniere usa per spiegare come effettuarla saranno nomi come "importo", "aliquota IVA", "totale", "codice fiscale", e così via, mentre un calcolatore, come apparecchiatura elettronica, "comprende" solo segnali elettrici corrispondenti, grossolanamente, al fatto che in un circuito passi o no corrente.
Il problema non è semplice, in quanto l'utilizzo di un calcolatore dipende proprio dal fatto che sia facile e facilmente apprendibile costruire programmi da comunicargli.

Tra la formulazione di un problema nei termini della normale comunicazione umana e la realizzazione di un procedimento risolutivo su calcolatore (formulato nel "linguaggio" comprensibile alla macchina) vi è una lacuna, colmata da un processo in più tappe, schematizzato qui e nelle due pagine seguenti. Dalla formulazione del problema si passa, qui, alla stesura di un algoritmo per la sua risoluzione e quindi alla traduzione dell'algoritmo in un linguaggio di programmazione di alto livello.

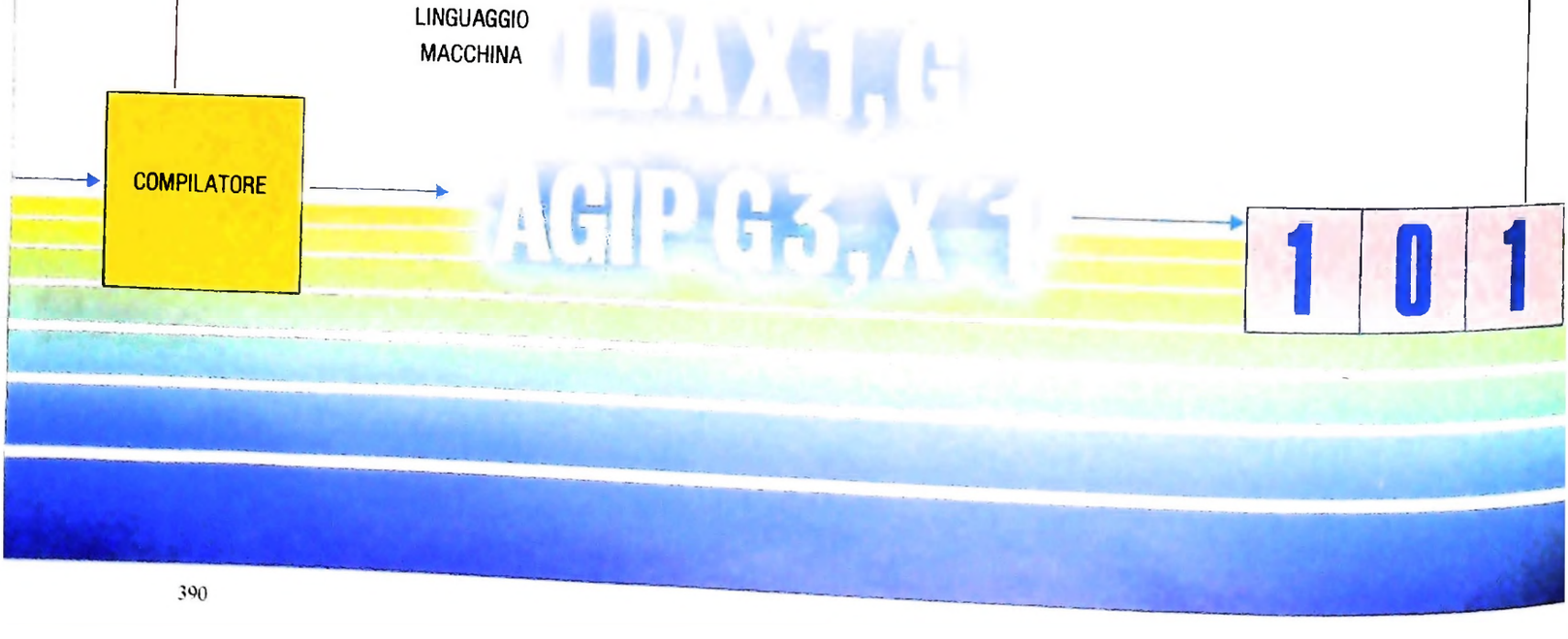


Sono per questo motivo nati i LINGUAGGI DI PROGRAMMAZIONE (cioè linguaggi che permettono al programmatore di scrivere gli algoritmi in una forma che sia consona al problema da risolvere), accompagnati da un adeguato insieme di supporti che consenta la loro trasformazione nel "linguaggio" proprio del calcolatore, fatto di segnali elettrici.
La figura della pagina precedente illustra la "catena" di passaggi necessari per procedere dal problema al programma eseguibile dal calcolatore.
Innanzitutto, viene identificato e delimitato il problema; quindi, viene costruito un algoritmo che costituisca la descrizione delle successioni di operazioni da compiere per risolverlo; infine, si provvede a realizzare una versione dell'algoritmo in un preciso linguaggio di programmazione (come per esempio il BASIC) ottenendo un programma.
Tutte queste operazioni richiedono capacità critica e inventiva, e sono effettuate dall'uomo.
A questo punto, interviene la fase di comunicazione al calcolatore, con i seguenti passi: il programma scritto nel linguaggio di programmazione viene tradotto da un *compilatore* in un programma equivalente, ma in cui le istruzioni sono quelle elementari direttamente eseguibili da un calcolatore (per esempio, un'istruzione BASIC come 10 LET A=B+C viene tradotta tipicamente in una successione di tre istruzioni che potrebbero suonare come:
"carica il contenuto della cella di memoria B in uno speciale 'accumulatore' ";
"somma all''accumulatore' il contenuto della cella C";
"ricopia il contenuto dell'accumulatore nella cella A").

Il programma così tradotto quindi viene trasformato in una successione di "zeri" e "uni" (che corrispondono concettualmente alla notazione di "circuito aperto: non passa corrente" e "circuito chiuso: passa corrente"), che corrispondono alla codifica delle istruzioni, così come i circuiti del calcolatore sono in grado di eseguirle (per esempio, l'operazione "somma all'accumulatore" potrebbe essere codificata come "10110" e "la cella B" potrebbe essere codificata come "0000001"); tali "zeri" e "uni" sono poi di fatto la rappresentazione simbolica che noi usiamo per denotare l'assenza o la presenza dei segnali elettrici.
Sarà proprio quest'ultima la forma in cui il programma verrà effettivamente eseguito dal calcolatore.
Normalmente, i passaggi logici di traduzione descritti sono svolti in modo più abbreviato: in genere il compilatore fornisce direttamente la codifica del programma sotto forma di "zeri" e "uni".

## Il compilatore

Dalla descrizione fatta potrebbe sembrare che un compilatore sia una strana apparecchiatura che effettui la traduzione; in realtà il problema della traduzione è un problema di elaborazione di informazione, né più né meno come quello di calcolare una fattura, e il modo più semplice di realizzare un compilatore è quello di farne un programma; avremo così un programma compilatore per ogni linguaggio di programmazione che vogliamo adottare.



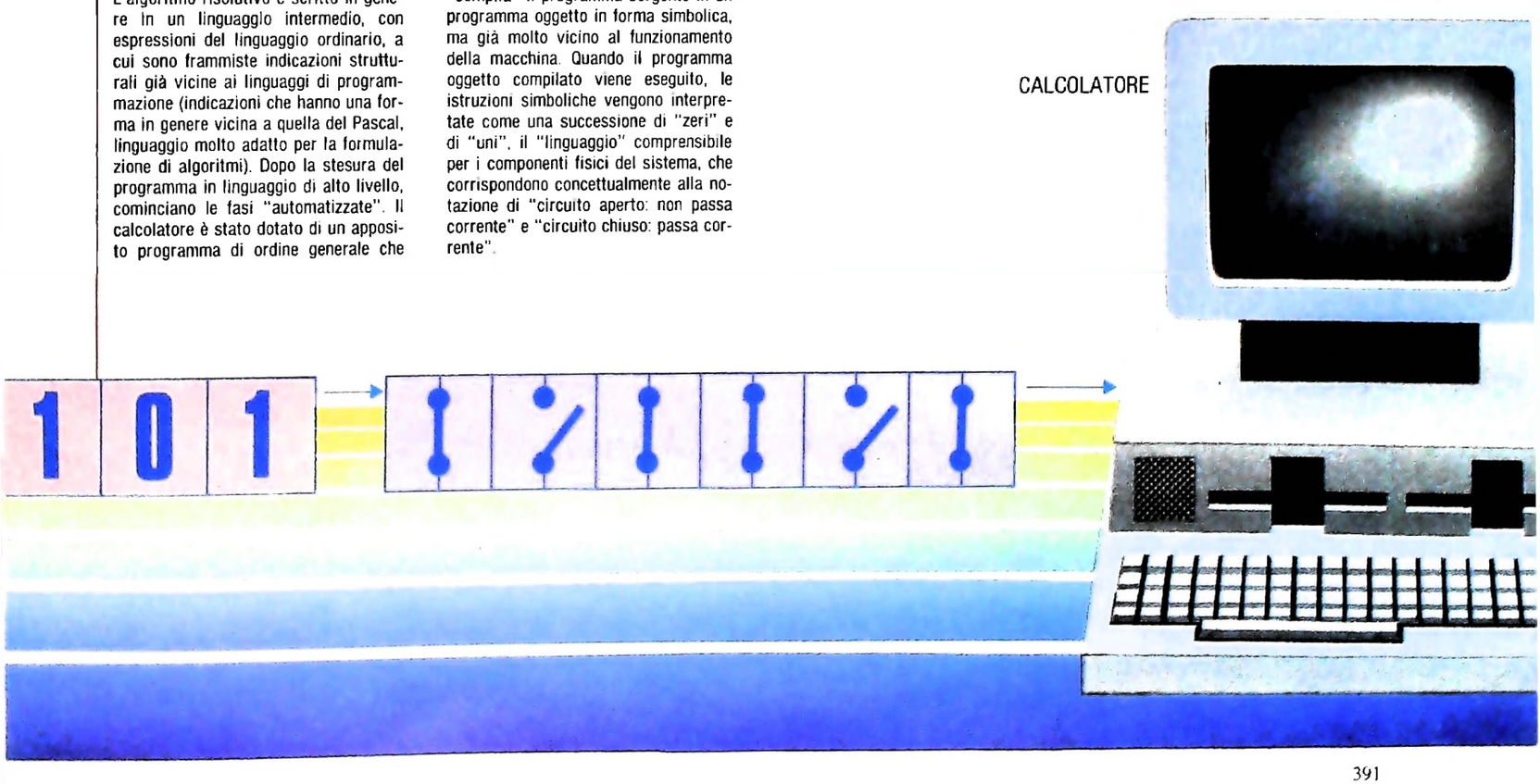
## Gli interpreti

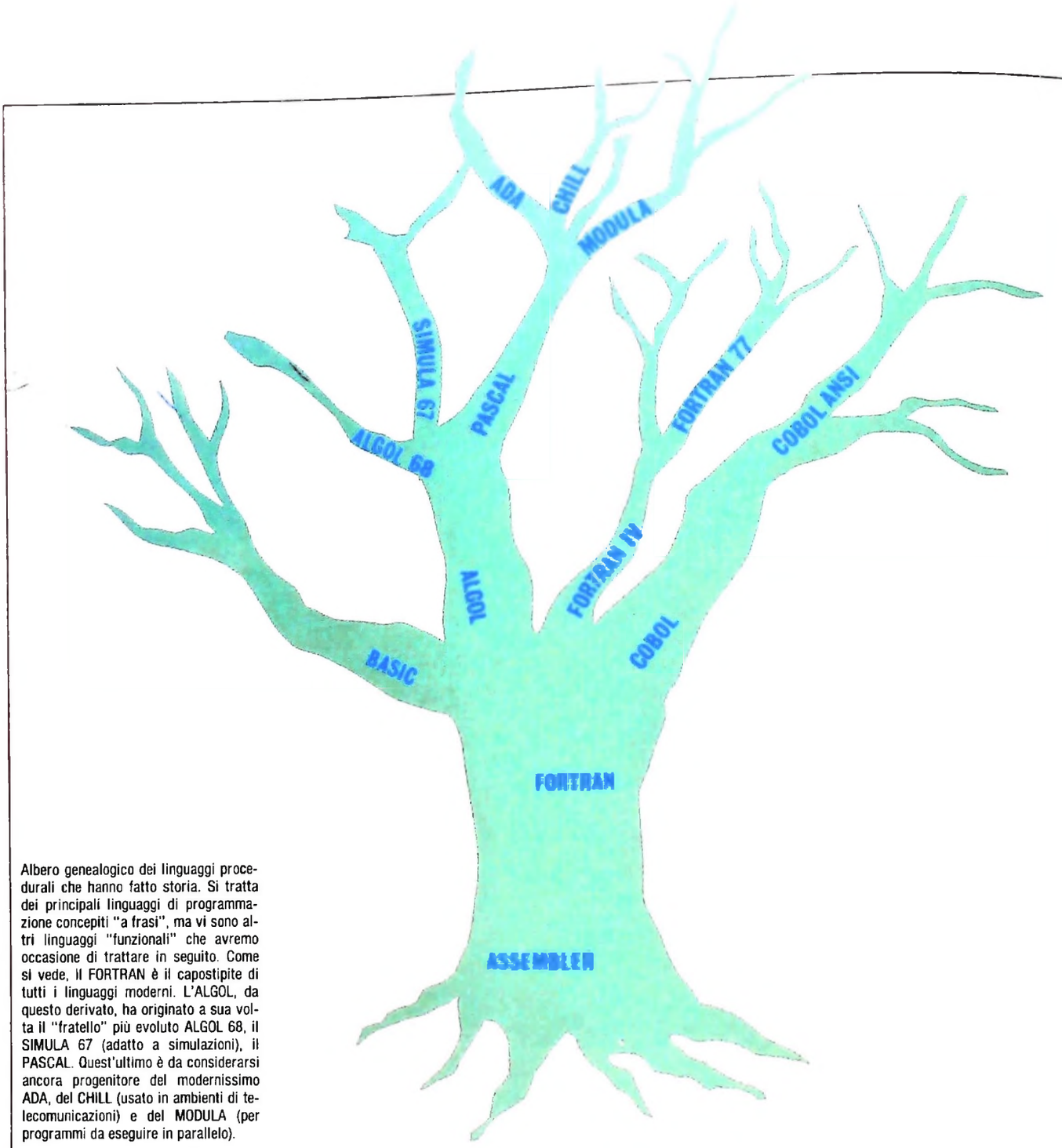
In taluni casi, i programmi scritti dal programmatore non vengono tradotti da un compilatore per essere eseguiti, ma subiscono un trattamento completamente diverso: è pensabile infatti di costruire *programmi interpreti* che svolgono la seguente funzione: leggono il programma scritto dal programmatore, istruzione per istruzione, analizzano ogni istruzione, ne comprendono il senso e la eseguono operando su variabili da loro stessi predisposte. Tale è proprio il caso del linguaggio BASIC sul calcolatore M10 e sulla maggior parte dei "personal" e "home" computer.
In genere gli interpreti hanno la caratteristica di facilitare le operazioni (non è necessario innescare la fase di compilazione), di essere più semplici da realizzare e di non richiedere molta memoria per funzionare; per contro l'esecuzione di un programma interpretato è inevitabilmente più lenta che non se lo stesso programma venisse tradotto e poi eseguito direttamente dal calcolatore.

## Le gerarchie di linguaggi di programmazione

All'inizio della storia dei calcolatori, quando non erano a disposizione compilatori, era necessario scrivere i programmi direttamente nella forma comprensibile dal calcolatore (non come sequenza di 0 e 1, ma in una forma intermedia con numeri scritti, ad esempio, in base ottale, cosicché una istruzione poteva avere l'aspetto della sequenza 065771; ma con quanta fatica e quanta difficoltà!).
Il passo successivo è stato quello di mettere a disposizione linguaggi in cui a ogni istruzione eseguibile dal calcolatore corrisponda una frase del linguaggio, come una forma mnemonica che faciliti l'uso (per esempio ADD B per dire di sommare il contenuto della variabile B all'accumulatore): si parla in questo caso di *linguaggio assembler*.
Un linguaggio assembler è completamente legato alla struttura circuitale elettronica del calcolatore, cosicché, in genere, due calcolatori diversi dispongono di linguaggi assembler completamente diversi.
L'uso di linguaggi assembler è allo stato attuale molto limitato, in quanto la programmazione ne risulta estremamente complessa e difficile; tuttavia, la possibilità che l'assembler offre di poter controllare con i programmi anche le caratteristiche tecniche più in dettaglio dell'hardware del calcolatore, lo fa adottare in casi in cui sia molto importante la velocità di elaborazione e la visibilità di tali caratteristiche: per esempio, viene spesso usato il linguaggio assembler nella scrittura di quelle parti di programmi che pilotano la lettura di segnali su una linea telefonica, o i comandi di un registratore a cassette, e così via.
La disponibilità di programmi "traduttori" (gli *assemblatori*) per i linguaggi assembler ha permesso di scrivere più agevolmente altri compilatori e interpreti, per linguaggi più evoluti, come il BASIC. Questi a loro volta sono diventati candidati per nuove definizioni di linguaggi di programmazione e per la costruzione di compilatori sempre più raffinati.
Attualmente i compilatori vengono scritti quasi esclusiva-

L'algoritmo risolutivo è scritto in genere in un linguaggio intermedio, con espressioni del linguaggio ordinario, a cui sono frammiste indicazioni strutturali già vicine ai linguaggi di programmazione (indicazioni che hanno una forma in genere vicina a quella del Pascal, linguaggio molto adatto per la formulazione di algoritmi). Dopo la stesura del programma in linguaggio di alto livello, cominciano le fasi "automatizzate". Il calcolatore è stato dotato di un apposito programma di ordine generale che "compila" il programma sorgente in un programma oggetto in forma simbolica, ma già molto vicino al funzionamento della macchina. Quando il programma oggetto compilato viene eseguito, le istruzioni simboliche vengono interpretate come una successione di "zeri" e di "uni", il "linguaggio" comprensibile per i componenti fisici del sistema, che corrispondono concettualmente alla notazione di "circuito aperto: non passa corrente" e "circuito chiuso: passa corrente".






Albero genealogico dei linguaggi procedurali che hanno fatto storia. Si tratta dei principali linguaggi di programmazione concepiti "a frasi", ma vi sono altri linguaggi "funzionali" che avremo occasione di trattare in seguito. Come si vede, il FORTRAN è il capostipite di tutti i linguaggi moderni. L'ALGOL, da questo derivato, ha originato a sua volta il "fratello" più evoluto ALGOL 68, il SIMULA 67 (adatto a simulazioni), il PASCAL. Quest'ultimo è da considerarsi ancora progenitore del modernissimo ADA, del CHILL (usato in ambienti di telecomunicazioni) e del MODULA (per programmi da eseguire in parallelo).

mente in linguaggi evoluti, e anche quando un nuovo calcolatore viene progettato (quindi senza che su di esso siano disponibili assemblatori o compilatori per poterlo dotare di programmi) non è necessario seguire daccapo la trafila indicata: si sceglie infatti di costruire i compilatori e gli assemblatori per la nuova macchina su altre già esistenti.

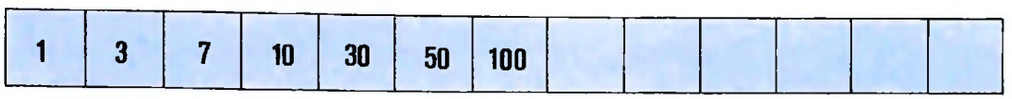
La storia dei linguaggi di programmazione non è tuttavia una storia fatta di evoluzioni tecnologiche, bensì essa è ricca di indagini dei meccanismi mentali dell'attività di programmazione e quindi di introduzione di strumenti linguistici sempre più evoluti e significativi dal punto di vista di tale attività. Descriveremo successivamente quali sono gli aspetti salienti di un linguaggio di programmazione, che permettano di classificarlo, di valutarlo, di confrontarlo con altri.

Successivamente, passeremo in rassegna un certo insieme di linguaggi, per esaminarne le caratteristiche e per illustrarne tipiche applicazioni. Tra questi esamineremo linguaggi Assembler; il Fortran (nato per effettuare calcolo scientifico e tecnico), il Cobol (particolarmente adatto ad applicazioni gestionali), il Pascal (adatto a qualunque tipo di applicazione e punto di riferimento culturale per i linguaggi di programmazione evoluti), il Lisp (tipico di applicazioni dell'Intelligenza Artificiale), Ada (ultimo arrivato e più evoluto della famiglia) e altri ancora.

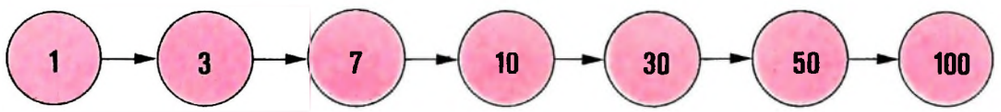
## Strutture di dati: liste

Abbiamo già visto come ordinare i dati in un array. Poniamoci adesso il problema di inserire un nuovo elemento in un array già ordinato: abbiamo due alternative:
- inserire il nuovo elemento in coda e ordinare nuovamente l'array;
- inserire il nuovo elemento direttamente nella posizione corretta.

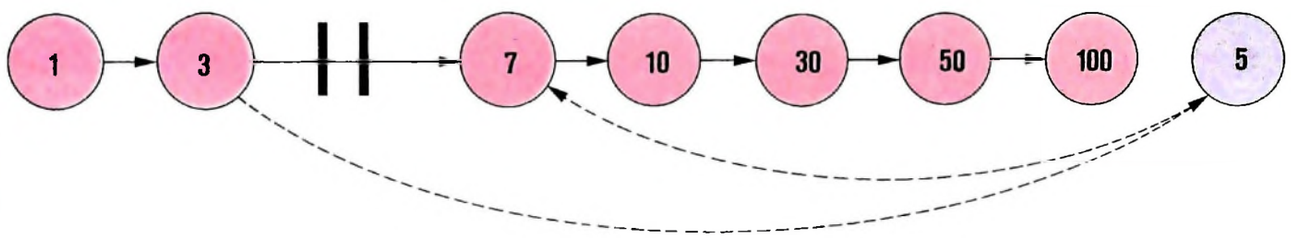
Proviamo a esaminare quest'ultima ipotesi.
Se l'array disponibile è per esempio il seguente:




e vogliamo inserire il valore 5, dovremmo porlo nella terza posizione. Ciò però implica che gli elementi successivi vengano fatti "slittare" a destra di una posizione. Questo metodo però risulterebbe particolarmente laborioso soprattutto quando il numero di elementi dell'array fosse elevato.
Proviamo allora una soluzione alternativa: immaginiamo che ogni elemento sia collegato al seguente da una "freccia", come nella seguente figura:




In tal caso, quando si voglia inserire un nuovo elemento sarà sufficiente "posizionare" opportunamente le frecce:




in modo che l'elemento inserito "punti" a quello immediatamente superiore e sia "puntato" da quello immediatamente inferiore.
In tal modo, in realtà inseriamo ugualmente l'elemento in coda agli altri valori, ma posizioniamo le frecce in modo da ricostruire l'ordine logico. Quello che otteniamo così è di avere un ordinamento fisico dei dati differente da quello logico.
In PASCAL una struttura dati di questo tipo si descrive così:

```
TYPE ELEMENTO = RECORD
        VALORE: INTEGER;
        PROSSIMO: FRECCIA
            END;
```

dove il tipo FRECCIA è definito come:

```
TYPE FRECCIA = ↑ ELEMENTO;
```

il significato di quanto sopra scritto è che si tratta di una struttura dati i cui elementi sono record costituiti da:
• un valore (in questo caso intero);
• una "freccia" che indica qual è l'elemento seguente.
Il campo PROSSIMO è di tipo FRECCIA, che è definito come un "puntatore" a una variabile di tipo ELEMENTO. In altre parole, stiamo descrivendo una struttura dati i cui elementi sono costituiti da un singolo valore e da un "indicatore" all'elemento successivo.

## Strutture statiche e strutture dinamiche

Normalmente, questo tipo di strutture dati viene generato elemento per elemento, via via se ne presenti l'esigenza. Fino ad ora avevamo visto strutture di dati che, comunque fossero usate, venivano allocate alla dimensione richiesta e da quel momento in avanti non potevano subire variazioni dimensionali. Tali strutture vengono dette "statiche" poiché non è possibile variarne le dimensioni una volta che siano state allocate. Quindi non è possibile:
• eliminare gli elementi in eccedenza, nel caso in cui non tutta la struttura sia stata usata;
• ottenere elementi aggiuntivi, quando quelli inizialmente allocati fossero stati interamente occupati.
Il limite delle strutture di dati statiche consiste nel fatto che richiedono una conoscenza a priori del numero di informazioni da trattare. In alternativa, quando il numero di tali informazioni non fosse noto, è necessario definire un numero massimo di informazioni che si intende trattare. Nel caso in cui il numero di valori risultasse minore, sarebbe comunque stato allocato il numero massimo di elementi con evidente spreco di spazio.
Nel caso invece in cui i valori da trattare fossero in realtà più del previsto, ci si troverebbe costretti a eliminare gli elementi eccedenti. Questo tipo di problema si risolve con le cosiddette strutture di dati "dinamiche", che consentono di generare elemento per elemento via via si presentino valori da trattare.
L'unico limite in questo caso è la quantità di memoria disponibile.
La struttura sopra descritta viene chiamata "lista" ed è un esempio tipico di struttura dinamica. Queste strutture richiedono una gestione della memoria complessa e sono pertanto disponibili solo nei linguaggi di programmazione più evoluti.
In PASCAL è disponibile anche una "primitiva", cioè un sottoprogramma direttamente richiamabile senza bisogno di definizione (la definizione è infatti implicita nel linguaggio), che consente di generare un nuovo elemento di una struttura dinamica. Tale primitiva è chiamata NEW.
Vediamo ora come procedere per generare un nuovo elemento di una struttura dinamica. Useremo innanzitutto una variabile che dichiareremo così:

```
VAR INFORMAZIONE: FRECCIA;
```

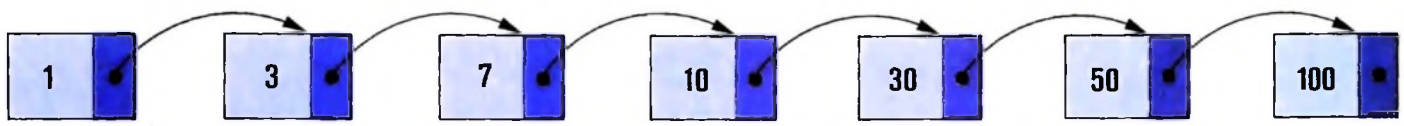
Procediamo dunque nel modo seguente:

```
PROCEDURE GENERA;
    BEGIN
        NEW (INFORMAZIONE);
        INFORMAZIONE. VALORE: = NUOVOVALORE
    END;
```

Il sottoprogramma descritto (si noti l'uso della sintassi di definizione di sottoprogrammi del PASCAL) ha l'effetto di generare tramite la primitiva NEW un nuovo record di tipo ELEMENTO. L'istruzione successiva assegna il nuovo valore al campo VALORE del nuovo record generato.

## Il costruttore di tipo "pointer"

Nel paragrafo precedente abbiamo visto una definizione di tipo alquanto particolare. Si tratta della definizione del tipo FRECCIA. Tale tipo è stato definito come un "puntatore" (indicato sintatticamente con il simbolo ↑) a una variabile di tipo elemento. Intuitivamente possiamo immaginarlo come una freccia che è "puntata" nella direzione dell'elemento successivo. In realtà tale "freccia" è l'indirizzo di memoria in cui il nuovo elemento è stato generato.
Si noti che il tipo ELEMENTO contiene, oltre al campo VALORE, un campo di nome PROSSIMO che rappresenta il "puntatore" all'elemento successivo. Si realizza così una struttura composta di elementi ciascuno dei quali "indica" il successivo. L'ultimo elemento della lista ha un puntatore vuoto, che cioè non punta a nulla. Possiamo rappresentare la struttura così costituita nel modo seguente:




In inglese questo tipo di strutture viene detto "linked" (in italiano "legato"), facendo riferimento al "link" ovvero al legame tra un componente e l'altro.
Quello così introdotto è un nuovo costruttore di tipo, che viene chiamato "pointer" (puntatore). Un puntatore in Pascal è lo strumento per generare dinamicamente una variabile in memoria.
La generazione avviene come abbiamo visto con la primitiva NEW che ha l'effetto:
- di generare la nuova variabile "puntata", cioè di allocare lo spazio di memoria necessario secondo il tipo di tale variabile;
- di "scrivere" l'indirizzo di tale area nella variabile puntatore indicata tra parentesi.

Alla nuova variabile potremo quindi assegnare i valori desiderati.
La variabile "puntata" può essere di tipo semplice o di tipo composto. È particolarmente frequente l'uso di tipi puntatore che indirizzano variabili di tipo record che a loro volta contengono un campo puntatore a una variabile dello stesso tipo, per costruire strutture dinamiche.

## Inserimento ordinato in una lista

Torniamo ora al problema iniziale di inserire un nuovo elemento in una struttura ordinata. Definiti i tipi ELEMENTO e FRECCIA, il sottoprogramma (in PASCAL detto procedura) si presenta così (con qualche semplificazione rispetto a un uso rigoroso del linguaggio):

```
PROCEDURE INSERIMENTORDINATO;
VAR LINK, L: FRECCIA;
BEGIN
    LINK: = puntatoreprimoelementolista;
```

```
            WHILE nuovovalore <= LINK↑.VALORE  DO
                BEGIN

                    L: = LINK;
                    LINK: = LINK↑.PROSSIMO
                END;
            inseriscielemento
        END;
```

dove INSERISCIELEMENTO è ancora una procedura così definita:

```
        PROCEDURE INSERISCIELEMENTO;
        VAR L1: FRECCIA;
        BEGIN
                NEW (L↑.PROSSIMO);
                L1: = L↑.PROSSIMO;
                L1↑. VALORE: = nuovovalore;
                L1↑.PROSSIMO: = LINK
        END;
```

In realtà l'algoritmo mostrato risolve il problema dell'inserimento solo nel caso in cui la lista contenga un valore inferiore e uno superiore a quello da inserire, quando cioè l'inserimento avvenga in posizione intermedia. Affronteremo più attentamente il problema dell'inserimento nella prossima lezione, in particolare per quanto riguarda l'inserimento all'inizio della lista e in coda.

La procedura INSERIMENTORDINATO effettua le seguenti operazioni:

- pone nella variabile puntatore di nome LINK l'indirizzo del primo elemento della lista, cioè la cosiddetta "radice" della struttura;
- scandisce gli elementi della lista fino a trovare la posizione in cui il nuovo elemento andrà inserito. La scansione avviene prelevando di volta in volta il puntatore di un elemento e controllando quindi che l'elemento così puntato sia maggiore di quello da inserire; si tiene anche memoria del puntatore all'elemento precedente. Così uscendo dall'iterazione saranno noti i puntatori all'elemento immediatamente inferiore e superiore a quello da inserire.
- inserisce quindi l'elemento richiamando la procedura INSERISCIELEMENTO. Quest'ultima dispone del puntatore all'elemento immediatamente superiore a quello da inserire. Per l'inserimento procede dunque così:
- con l'istruzione NEW (L↑. PROSSIMO) genera il nuovo elemento cioè riserva un'area di memoria per il nuovo elemento. L'indirizzo di tale area di memoria è posta nel campo puntatore dell'elemento immediatamente inferiore; L infatti è la variabile che contiene l'indirizzo dell'elemento inferiore;
- "ricorda" in L1 l'indirizzo del nuovo elemento generato;
- pone nel campo valore dell'elemento generato il nuovo valore;
- pone nel campo puntatore dell'elemento generato l'indirizzo dell'elemento ad esso superiore. Tale indirizzo era stato memorizzato nella variabile LINK.

## Cosa abbiamo imparato

In questa lezione abbiamo visto:

- il concetto di struttura statica e di struttura dinamica;
- il concetto di lista;
- l'uso di strutture dinamiche in PASCAL con la "primitiva" NEW;
- il costruttore di tipo POINTER.

# IL SISTEMA PER L'ELABORAZIONE MUSICALE (I)

## L'"ambiente" per l'elaborazione automatica adatto a ospitare le applicazioni di informatica musicale.

In precedenza abbiamo discusso dei principali concetti musicali in rapporto all'uso degli elaboratori elettronici (e in particolare dei personal/portable computer) per applicazioni musicali.
Qui vogliamo invece cominciare a parlare dell'*ambiente* per l'elaborazione automatica adatto a ospitare le applicazioni di informatica musicale. Discuteremo quindi del *sistema per l'elaborazione musicale* (SEM), delle sue funzionalità e della sua architettura, dell'interazione musicista/elaboratore e degli aspetti tecnici più rilevanti connessi con l'elaborazione numerica dei fenomeni musicali.

### Funzionalità del SEM

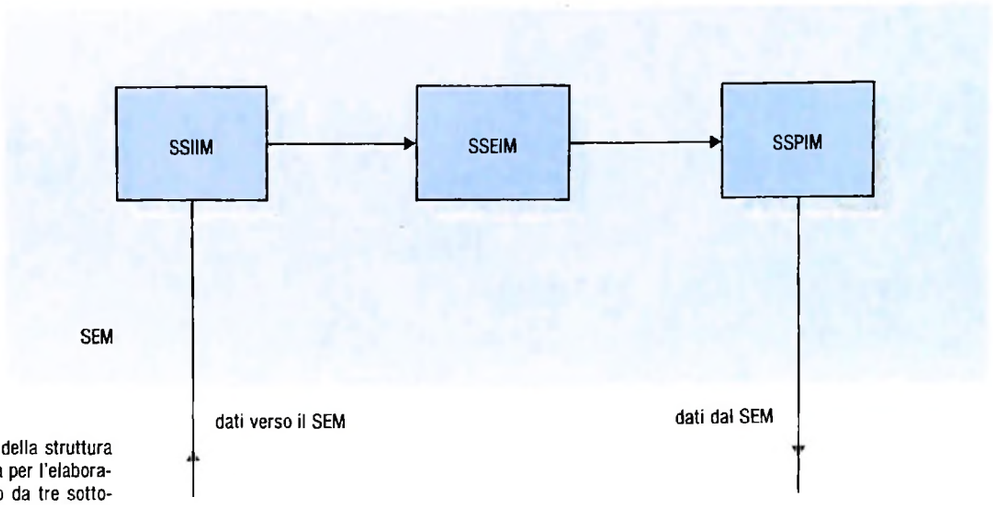
Abbiamo già accennato, parlando del concetto musicale di nota, alle funzionalità che caratterizzano un sistema capace di elaborare i fenomeni musicali: analisi, elaborazione e sintesi di testi musicali e analisi, elaborazione e sintesi di suoni.
Abbiamo anche accennato ai differenti livelli di rappresentazione dell'informazione musicale a cui possiamo operare mediante un elaboratore (livello strutturale ovvero della forma musicale, livello simbolico ovvero della partitura tipo pentagramma, livello fisico ovvero dei modelli matematici dei fenomeni acustici, infine livello operativo ovvero delle modalità realizzative).
I sei tipi di funzionalità distinti e i quattro differenti livelli di rappresentazione sono gli *ingredienti* della ricetta per la progettazione di un SEM; il SEM più generale li prevede tutti, ma praticamente succede che, a seconda della funzione prevalente a cui è destinato un SEM, consideriamo una ricetta particolare adatta alla specifica applicazione.
Così, come pensiamo ad un sistema per l'analisi del suono, non ci interessano i livelli strutturale e simbolico né le funzionalità riguardanti il testo, cioè le partiture; oppure, se pensiamo a un sistema per la stampa di partiture, non ci interessano i livelli fisico e operativo, né le funzionalità relative al trattamento del suono.
In dipendenza dalle caratteristiche del SEM di nostro interesse individuiamo allora l'opportuna ricetta di livelli di rappresentazione necessari e di funzionalità da implementare.
In qualche caso avremo funzionalità opzionali, cioè opportune ma non strettamente necessarie; ad esempio, un sistema orientato alla composizione musicale è bene preveda la possibilità dell'immediata verifica sonora della composizione in corso di sviluppo, anche se questo non può essere considerato essenziale; i compositori sono infatti spesso costretti a comporre "astrattamente", spesso non riescono a far eseguire le loro composizioni prima di mesi o anni dalla stesura della partitura; l'elaboratore può assolvere alla funzione, che sovente era stata affidata al pianoforte (in modo molto più completo poiché l'elaboratore può produrre tutti i timbri), di verifica sonora delle strutture musicali composte.
In ogni caso, in corrispondenza con gli ingredienti della ricetta del SEM definiamo l'insieme di *moduli* hardware e software di cui sarà costituito il SEM che abbiamo progettato.

### Architettura del SEM

A seconda dei dispositivi che costituiscono il SEM, dei flussi dell'informazione tra i dispositivi, delle funzionalità e, in generale, di come organizziamo la comunicazione e l'interazione tra le diverse componenti fisiche e logiche del SEM, abbiamo diverse architetture del SEM.
Se consideriamo un SEM come costituito da un sottosistema per l'introduzione delle informazioni musicali (SSIIM), da un sottosistema per l'elaborazione dell'informazione musicale (SSEIM) e da un sottosistema per la produzione di informazioni musicali (SSPIM), possiamo caratterizzare i tipi di dati connessi con il SSIIM e il SSPIM e i dispositivi che costituiscono il SSEIM.

Gli ingressi del SSIIM possono essere:
a) dati da tastiera alfanumerica;
b) dati da tastiera musicale;
c) dati da memorie magnetiche;
d) dati da dispositivi analogici di vario tipo;
e) dati da dispositivi particolari (pedali, light-pen ecc.).

Le uscite del SSPIM possono essere:
a) dati a terminale video;
b) dati a terminali stampanti;
c) dati a memorie magnetiche;




Sopra, schematizzazione della struttura ad alto livello del sistema per l'elaborazione musicale, costituito da tre sottosistemi. Gli schemi architetturali di tale sistema (il SEM) sono indicati qui sotto.

## Schemi architetturali di SEM

architettura di dispositivi analogici:

analizzatore analogico → processore analogico → sintetizzatore analogico

architettura ibrida:

analizzatore analogico → processore digitale (del testo musicale) → sintetizzatore analogico

architettura di dispositivi digitali per sintesi diretta:

convertitore analogico/digitale → processore digitale → convertitore digitale/analogico

architettura mista di dispositivi digitali:

processore digitale (del testo musicale) ↔ processore digitale veloce (del suono) ↔ convertitori A/D e D/A

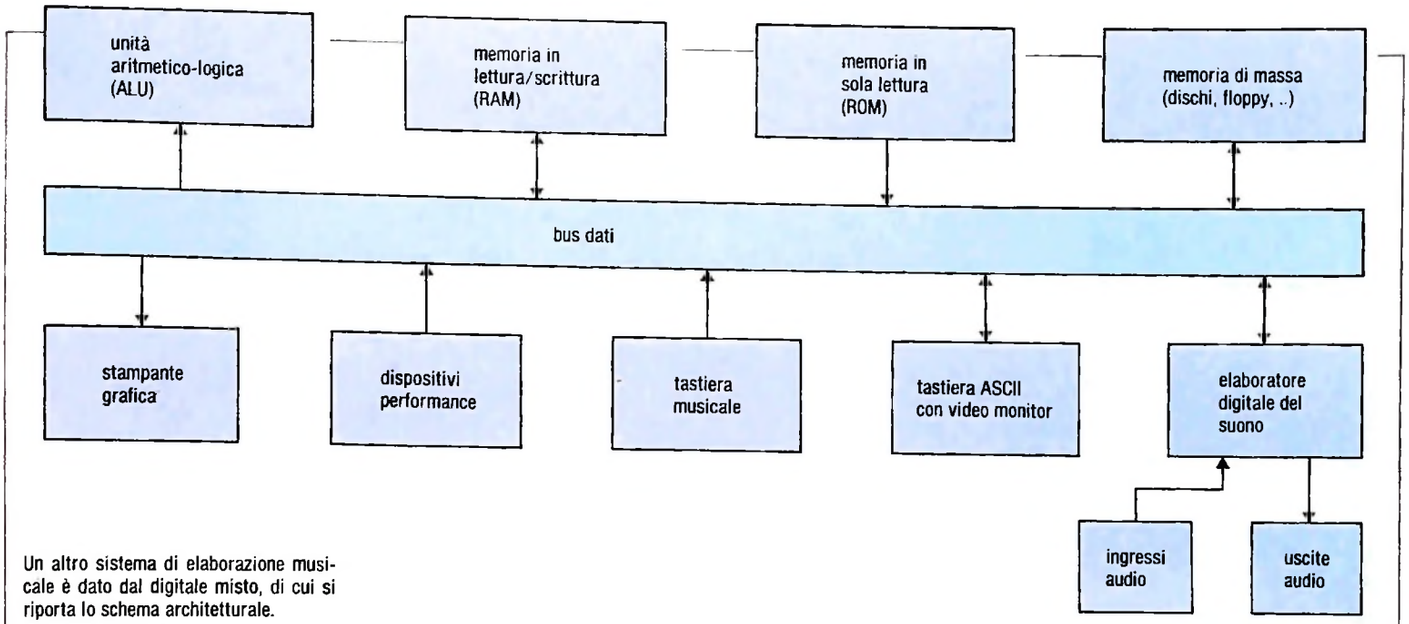
architettura digitale multiprocessore:

gerarchica:

processore principale → processore-1
processore principale → processore-2
processore principale → processore-3

parallela:

processore-1 ↕ processore-2 ↕ processore-3



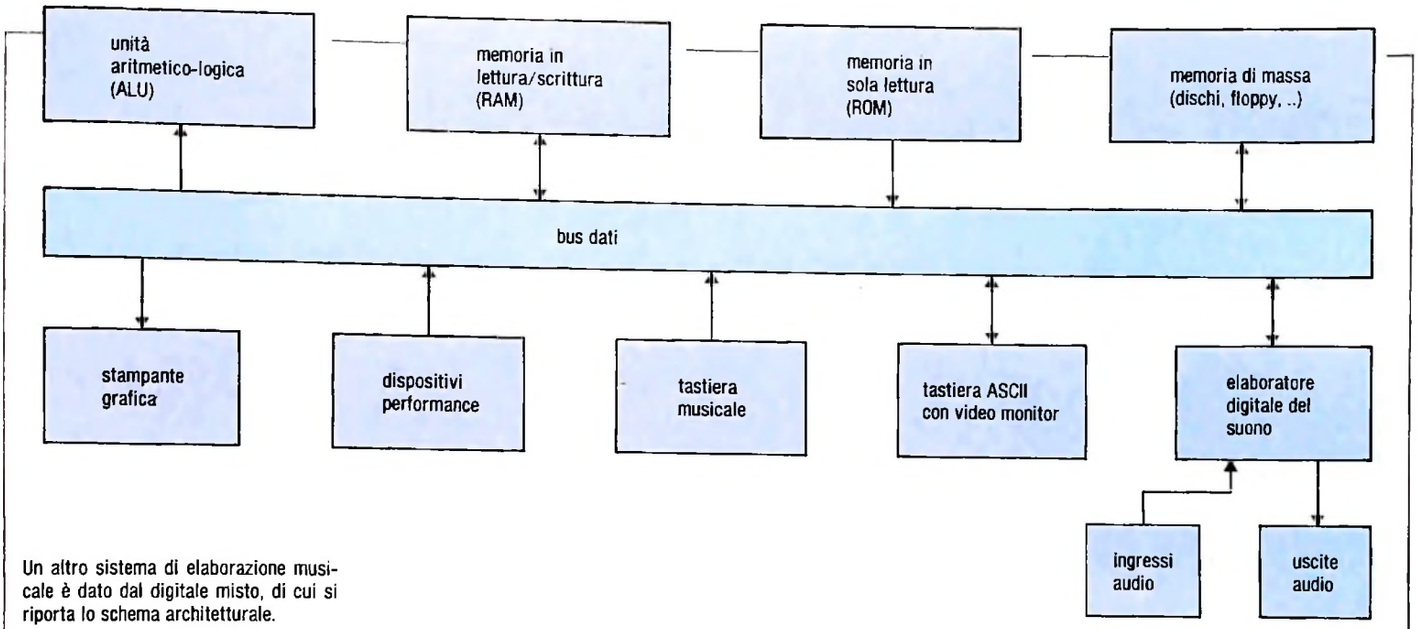

Un altro sistema di elaborazione musicale è dato dal digitale misto, di cui si riporta lo schema architetturale.

d) dati analogici di vario tipo;
e) dati a dispositivi particolari (luci, filtri, riverberatori);
f) dati a uscite grafiche (printer/plotter, video ad alta risoluzione ecc.).

Il SSEIM può essere costituito da:
a) architettura di dispositivi *analogici* (sia per l'elaborazione del testo che, più tipicamente, per l'elaborazione del suono);
b) architettura *ibrida* (dispositivi analogici per l'elaborazione del suono e dispositivi digitali per l'elaborazione del testo);
c) architettura di dispositivi *digitali* per *sintesi diretta* (in cui un elaboratore "normale" viene utilizzato sia per il trattamento del testo che del suono e produce risultati in tempi piuttosto lunghi);
d) architettura *mista* di dispositivi *digitali* (in cui un elaboratore "normale" tratta il testo musicale e uno o più elaboratori "ultraveloci" trattano il suono);
e) architettura *digitale multiprocessore* (in cui si tende ad associare un elaboratore a ogni funzione musicale, come ad esempio il controllo di una tastiera musicale); in questo caso possiamo avere architetture *gerarchiche* (in cui un elaboratore controlla tutti gli altri) o *parallele* (in cui più elaboratori procedono concorrentemente sincronizzandosi mediante scambi di informazioni).
È chiaro che le diverse possibilità architetturali comportano diversi costi e l'impiego di tecnologie differenti.
Nella figura in alto in questa pagina è mostrato in dettaglio lo schema architetturale di un SEM di tipo generale (cioè capace di tutte le funzionalità che abbiamo previsto) basato su un modello architetturale digitale misto.

## Modalità di interazione

L'adozione di un'architettura funzionale piuttosto che un'altra porta a sostanziali differenziazioni dei SEM anche rispetto ai tempi di risposta del particolare SEM considerato; diciamo allora che un SEM opera in:
a) *tempo differito*, quando i tempi di risposta sono lunghi (nell'ordine anche delle ore);
b) *modalità interattiva*, quando i tempi di risposta sono brevi (nell'ordine di qualche minuto primo al più);
c) *tempo reale*, quando il SEM risponde immediatamente alle nostre richieste, per il trattamento del testo e del suono.
Vediamo quindi che un SEM in tempo differito richiede dispositivi meno costosi e sofisticati, ma non permette la performance musicale ed è meno comodo da usare degli altri SEM, sia per il musicologo che per il compositore.
I SEM interattivi permettono un'efficace realizzazione di qualsiasi attività musicale tranne la performance; tipicamente, richiedono risorse di calcolo con capacità di memoria e velocità di calcolo ben maggiori dei SEM in tempo differito.
I SEM in tempo reale sono ovviamente i più interessanti poiché non hanno limitazioni teoriche rispetto a quello che possono fare; per ora, però, hanno costi ancora troppo alti (intorno a qualche centinaio di milioni di lire) per poter essere acquisiti a livello di singolo individuo; richiedono infatti l'uso di tecnologie particolari attualmente molto costose.
Il problema cruciale, che determina questa classificazione rispetto ai tempi di risposta, è costituito dal trattamento numerico del suono: un segnale audio (e quindi un suono) di buona qualità richiede l'elaborazione di un flusso di informazioni pari a circa 1.000.000 di bit al secondo per canale audio; un segnale stereo richiede quindi due milioni di bit al secondo, un segnale quadrafonico ne richiede quattro.
Il trattamento dell'informazione relativa al testo è invece molto meno oneroso: sono necessarie quantità di informazioni dell'ordine di qualche migliaio di bit al secondo.
Per questi motivi i piccoli sistemi di elaborazione (personal computer e simili) sono molto indicati per il trattamento del testo musicale e sono invece del tutto inadeguati per il trattamento del suono a un buon livello di qualità audio.




Un altro sistema di elaborazione musicale è dato dal digitale misto, di cui si riporta lo schema architetturale.

d) dati analogici di vario tipo;
e) dati a dispositivi particolari (luci, filtri, riverberatori);
f) dati a uscite grafiche (printer/plotter, video ad alta risoluzione ecc.).

Il SSEIM può essere costituito da:
a) architettura di dispositivi *analogici* (sia per l'elaborazione del testo che, più tipicamente, per l'elaborazione del suono);
b) architettura *ibrida* (dispositivi analogici per l'elaborazione del suono e dispositivi digitali per l'elaborazione del testo);
c) architettura di dispositivi *digitali* per *sintesi diretta* (in cui un elaboratore "normale" viene utilizzato sia per il trattamento del testo che del suono e produce risultati in tempi piuttosto lunghi);
d) architettura *mista* di dispositivi *digitali* (in cui un elaboratore "normale" tratta il testo musicale e uno o più elaboratori "ultraveloci" trattano il suono);
e) architettura *digitale multiprocessore* (in cui si tende ad associare un elaboratore a ogni funzione musicale, come ad esempio il controllo di una tastiera musicale); in questo caso possiamo avere architetture *gerarchiche* (in cui un elaboratore controlla tutti gli altri) o *parallele* (in cui più elaboratori procedono concorrentemente sincronizzandosi mediante scambi di informazioni).
È chiaro che le diverse possibilità architetturali comportano diversi costi e l'impiego di tecnologie differenti.
Nella figura in alto in questa pagina è mostrato in dettaglio lo schema architetturale di un SEM di tipo generale (cioè capace di tutte le funzionalità che abbiamo previsto) basato su un modello architetturale digitale misto.

## Modalità di interazione

L'adozione di un'architettura funzionale piuttosto che un'altra porta a sostanziali differenziazioni dei SEM anche rispetto ai tempi di risposta del particolare SEM considerato; diciamo allora che un SEM opera in:
a) *tempo differito*, quando i tempi di risposta sono lunghi (nell'ordine anche delle ore);
b) *modalità interattiva*, quando i tempi di risposta sono brevi (nell'ordine di qualche minuto primo al più);
c) *tempo reale*, quando il SEM risponde immediatamente alle nostre richieste, per il trattamento del testo e del suono.
Vediamo quindi che un SEM in tempo differito richiede dispositivi meno costosi e sofisticati, ma non permette la performance musicale ed è meno comodo da usare degli altri SEM, sia per il musicologo che per il compositore.
I SEM interattivi permettono un'efficace realizzazione di qualsiasi attività musicale tranne la performance; tipicamente, richiedono risorse di calcolo con capacità di memoria e velocità di calcolo ben maggiori dei SEM in tempo differito.
I SEM in tempo reale sono ovviamente i più interessanti poiché non hanno limitazioni teoriche rispetto a quello che possono fare; per ora, però, hanno costi ancora troppo alti (intorno a qualche centinaio di milioni di lire) per poter essere acquisiti a livello di singolo individuo; richiedono infatti l'uso di tecnologie particolari attualmente molto costose.
Il problema cruciale, che determina questa classificazione rispetto ai tempi di risposta, è costituito dal trattamento numerico del suono: un segnale audio (e quindi un suono) di buona qualità richiede l'elaborazione di un flusso di informazioni pari a circa 1.000.000 di bit al secondo per canale audio: un segnale stereo richiede quindi due milioni di bit al secondo, un segnale quadrafonico ne richiede quattro.
Il trattamento dell'informazione relativa al testo è invece molto meno oneroso: sono necessarie quantità di informazioni dell'ordine di qualche migliaio di bit al secondo.
Per questi motivi i piccoli sistemi di elaborazione (personal computer e simili) sono molto indicati per il trattamento del testo musicale e sono invece del tutto inadeguati per il trattamento del suono a un buon livello di qualità audio.

# HARDWARE GRAFICO: IL DIGITALIZZATORE

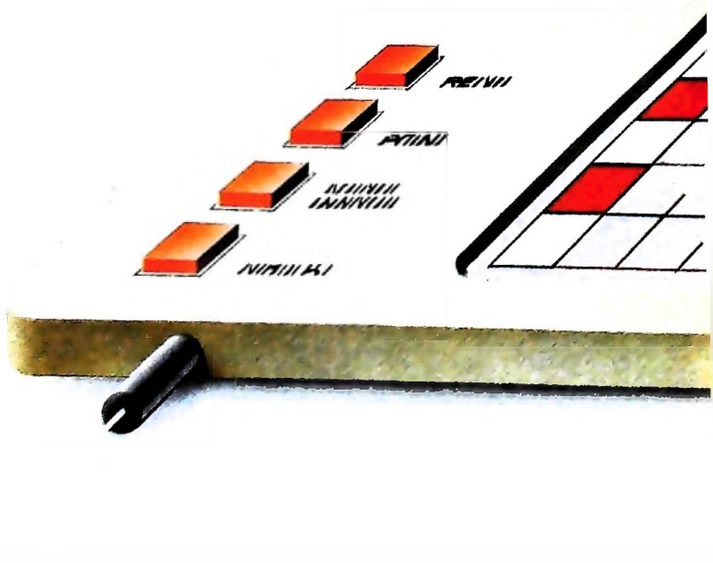
## Il digitalizzatore (digitizer) è un'unità di input che permette di convertire informazioni grafiche su carta in una forma numerica intelligibile al calcolatore.

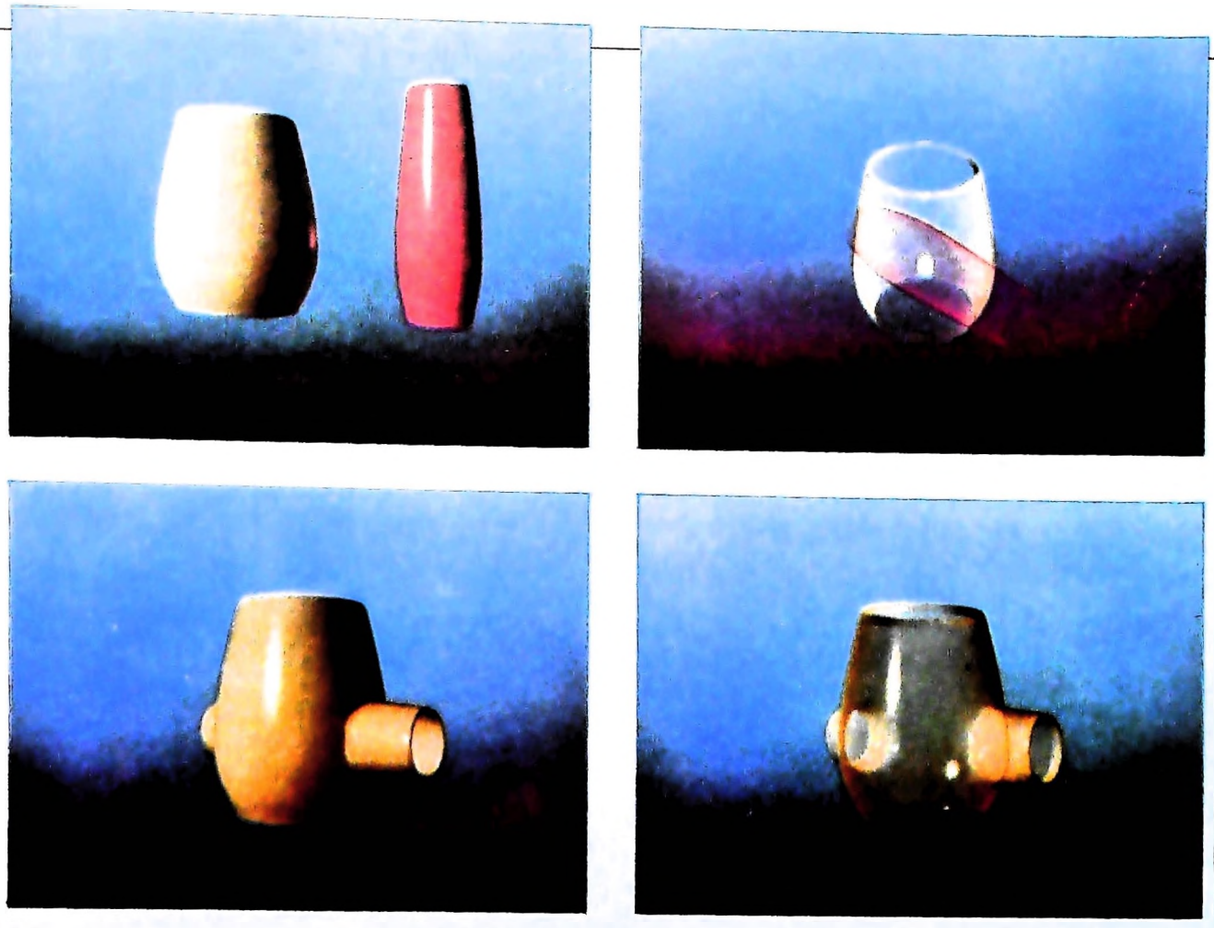
Abbiamo finora avuto modo di renderci conto di come la computergrafica si sia procurata un suo specifico spazio nel panorama dello sviluppo informatico. Da quando Ivan Sutherland, nell'ormai lontano 1963, mise a punto il suo SKETCHPAD, il primo vero sistema con capacità grafiche, molte sono state le applicazioni della computergrafica negli ambiti più disparati.
In molti settori ingegneristici, gli schemi di impianti e le documentazioni tecniche in genere, sono automaticamente disegnate con plotter digitali ad alta risoluzione grafica. I videogame, con i loro esplosivi display, costituiscono un'altra applicazione. Ancor più eloquenti sono i sistemi per la simulazione del volo aereo, attualmente utilizzati per l'addestramento dei piloti, mediante i quali viene rappresentata dinamicamente su schermo la vista dal punto di osservazione della cabina di pilotaggio. Un'ulteriore spettacolare applicazione è costituita dalla realizzazione di effetti speciali nella cinematografia. Anche in architettura la possibilità di riprodurre su video le volumetrie costituisce un passo avanti verso la realizzazione e lo studio di nuovi ambienti.
Per ottenere questi risultati sono stati necessari un sostanziale sviluppo tecnologico per la parte hardware e un enorme sforzo creativo per la parte software.

### Una prima classificazione

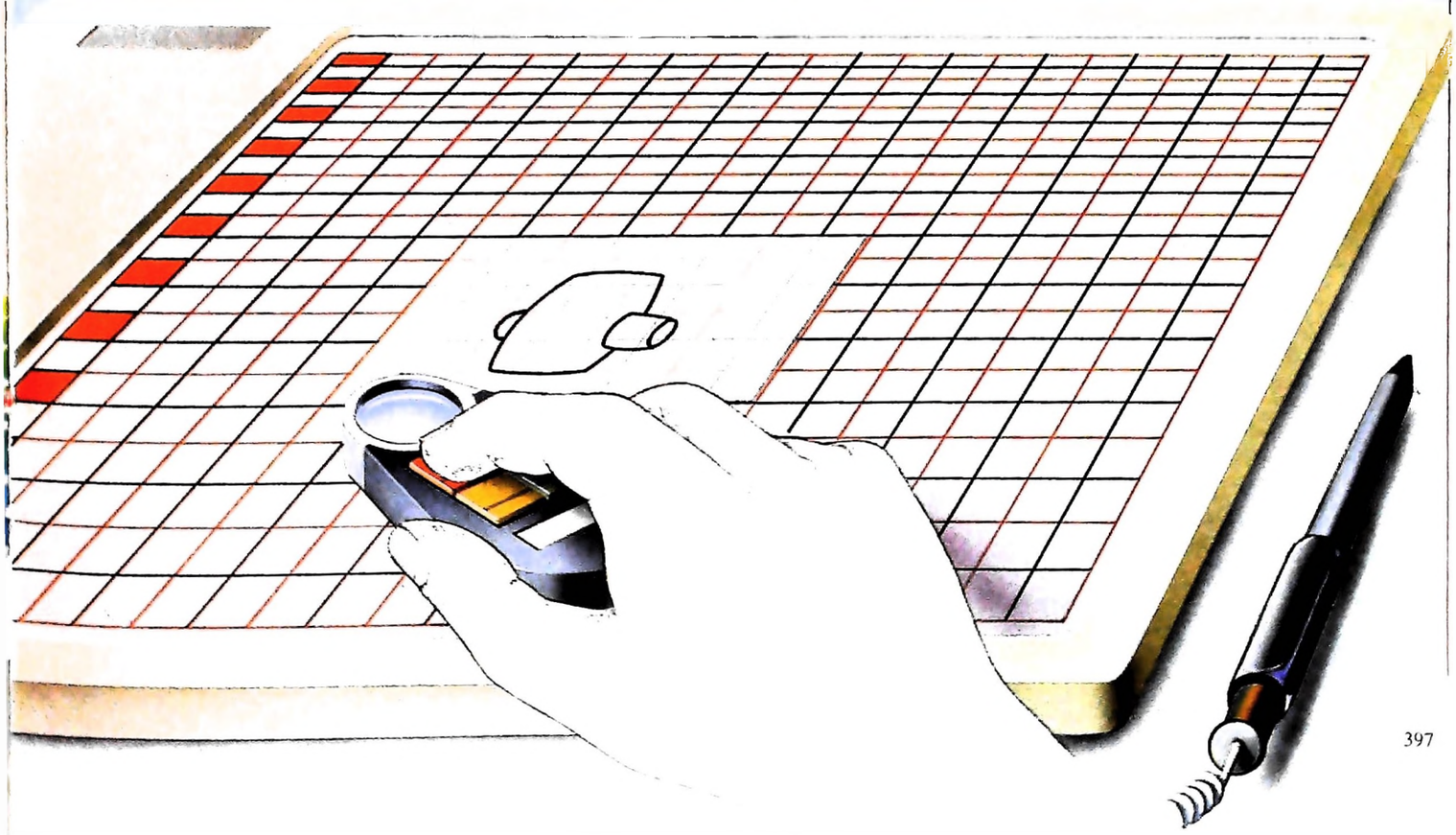
L'elaborazione di forme grafiche è attuabile con qualsiasi tipo di computer: dal personal al maxi. Ovviamente, però, la qualità e la complessità delle immagini ottenibili dipendono fortemente dalle potenzialità dello strumento, cioè dalla sua velocità di calcolo, dalla disponibilità di memoria e dal tipo di periferiche disponibili. Con un personal come M10, si possono ottenere grafici di funzioni, istogrammi, diagrammi, semplici disegni bidimensionali e anche qualche immagine tridimensionale; ma per ottenere risultati quali la modellazione di solidi in tre dimensioni, la realizzazione in maniera interattiva di progetti meccanici o la creazione di immagini con simulazioni di ombreggiature e di sorgenti luminose, è indispensabile fare ricorso a calcolatori molto più grossi e dotati di una tecnologia molto più sofisticata. Per poter effettuare elaborazioni grafiche è necessaria infatti, oltre al calcolatore vero e proprio, tutta una serie di apparecchiature (*device* in inglese) complementari. Un sistema grafico in configurazione media è generalmente costituito dalle seguenti apparecchiature: un terminale grafico, per visualizzare sullo schermo le immagini ottenute; un plotter, per riprodurre su carta o su altri supporti l'immagine; un'unità di input come il digitalizzatore (*digitizer* in inglese), che consente di convertire informazioni grafiche su carta in una forma numerica intelligibile al calcolatore.
Ovviamente in un sistema non sempre devono essere presenti tutte le apparecchiature grafiche elencate: per esempio, se non è necessario fornire i dati di input a partire da un'immagine riprodotta su carta, è inutile la presenza del digitizer, oppure, se non serve avere la riproduzione su carta, si può

Come si usa un digitalizzatore: un trasduttore, mosso manualmente, percorre la griglia di fili percorsi da corrente continua del digitalizzatore, consentendo così di rilevare le coordinate dei punti dell'immagine. Sopra, esempi di viste tridimensionali.





ACM - ARCHIVIO EIDOS



ACM - ARCHIVIO EIDOS

eliminare il plotter. Dato che il costo di queste apparecchiature è tutt'altro che trascurabile, è bene che l'acquirente di un sistema grafico valuti attentamente le sue esigenze.
Possiamo dividere l'hardware di un sistema grafico nelle seguenti categorie:
— unità di input;
— unità di interazione;
— unità di output.

## Le unità di input

In un sistema grafico non si può pensare di dover utilizzare come unità di input la sola tastiera, che ha caratteristiche tipicamente adatte all'inserimento alfanumerico dei dati (testi, programmi, elenchi di numeri). Se bisogna inserire nel calcolatore le informazioni relative a un'immagine già esistente su carta è assolutamente necessaria un'apparecchiatura che sia in grado di rilevare coppie di coordinate dal foglio di carta e convertirle in forma digitale per essere così comprese dall'elaboratore. Provate a pensare quanto tempo sarebbe necessario per rilevare manualmente le coordinate dei punti di una curva qualsiasi da un disegno.

## Il digitalizzatore

La più tipica unità di input che realizza questa attività è il digitalizzatore, simile ad un tecnigrafo, con una superficie di lavoro in materiale rigido, sotto cui è posta una griglia di conduttori elettrici che sono percorsi da corrente continua. Sulla superficie si fissa il foglio di carta su cui è riprodotto il disegno da elaborare e su di essa viene fatto scorrere manualmente un trasduttore magnetico, collegato al digitalizzatore e dotato di una lente a reticolo che fa da mirino. Quando si è posizionato il trasduttore esattamente sul punto prescelto del disegno, per convertirlo in forma digitale si preme un apposito pulsante: il trasduttore genera correnti indotte nel grigliato sottostante, le quali vengono rilevate da un sistema elettronico e memorizzate dal computer come coppia di coordinate relative al punto selezionato. Percorrendo il disegno punto per punto è quindi possibile trasferirlo nella memoria dell'elaboratore in forma digitale.
Il digitalizzatore consente però anche un'altra attività. La sua superficie piana è infatti suddivisa in due zone: una propriamente detta "area di lavoro", sulla quale avviene la conversione delle coordinate del disegno appena descritta, l'altra chiamata "area di menù", che consente un utilizzo ancor più sofisticato dello strumento.
Il concetto di "menù" è di fondamentale utilità nell'input grafico. Infatti una volta rilevate due coppie di coordinate, bisogna segnalare al computer se i due punti individuati vanno per esempio uniti da una linea retta, oppure se appartengono a una circonferenza o a qualche altra figura geometrica. Ecco che allora viene isolata un'area apposita sulla superficie del digitalizzatore, definita come "area di menù": essa è divisa in quadrati e le coordinate che stanno all'interno di ciascuno di essi non hanno il significato di coppie di valori numerici, ma di funzioni operative. In sostanza, l'operatore rileva semplicemente una di queste funzioni dal corrispondente quadrato del menù mediante il trasduttore magnetico, e la associa ad una o più coordinate del disegno. Questa operazione è molto rapida e semplice rispetto a quella convenzionale che utilizza la tastiera alfanumerica. Il risparmio di tempo fornito dalle funzioni di menù è immediatamente comprensibile: basta infatti pensare che per determinare le coordinate di un cerchio da un disegno, non sarà necessario rilevare punto per punto le coordinate della sua circonferenza,

bensì il solo centro, il raggio e la funzione di menù che genera il cerchio.
Le principali funzioni di menù che generalmente compaiono sul digitalizzatore sono:
— funzioni geometriche: punto, linea, arco, cerchio, rettangolo;
— trasformazioni geometriche: scala, rotazione, traslazione, immagine speculare;
— funzioni di disegno: selezione colori, spessore di linea, fusione e separazione di figure;
— funzioni di sistema: memorizzazione o richiamo di dati dall'archivio, lista dei comandi ecc.

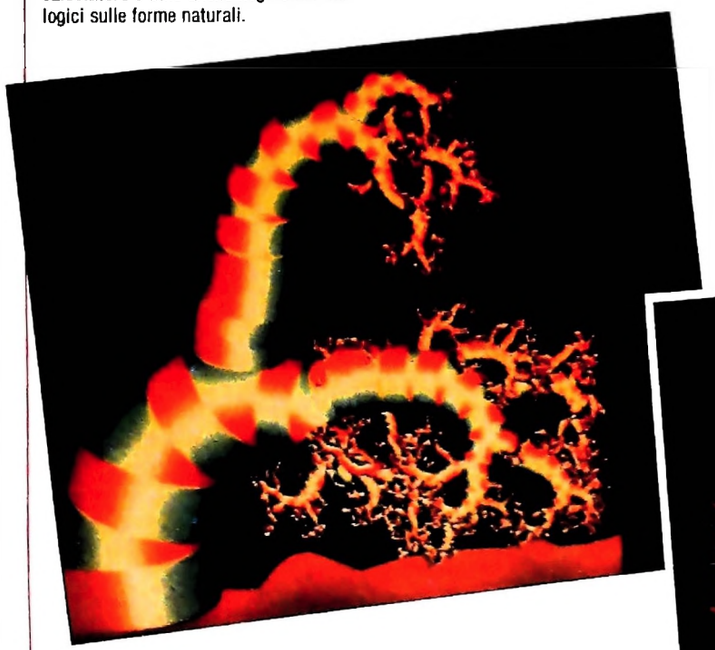
## Alcune caratteristiche

La risoluzione di un digitalizzatore, ossia la capacità di distinguere due punti vicini fra loro, in genere varia da 0,1 a 0,25 mm. Questo dato è soggetto però all'abilità dell'operatore nel posizionare correttamente il trasduttore.
La rapidità con cui è possibile rilevare sequenze di valori successivi varia dai 100 ai 200 punti al secondo. Questo parametro è importante quando si devono rilevare punti in modo continuo, per esempio da una curva.
Le possibilità grafiche del digitalizzatore consentono di realizzare anche disegni a mano libera: basta infatti utilizzare la superficie del digitalizzatore come se fosse un foglio di carta e il trasduttore come matita per inviare al computer le coordinate dei punti che costituiscono il disegno. Con l'ausilio poi delle funzioni di menù è possibile realizzare le linee di connessione fra i vari punti.
Le dimensioni dei digitalizzatori possono andare da un massimo di 120x150 cm circa a un minimo di 11x11 cm. Quelli di piccole dimensioni vengono anche chiamati "tavolette" (tablet o pad in inglese). La loro comodità e il loro costo, ormai accessibile anche all'utente privato, hanno reso insostituibile questo strumento, per chi ha esigenze di elaborazioni grafiche anche nel mercato dei personal computer.

## Le unità di output

Nella quasi totalità dei sistemi grafici esiste almeno un'unità di output. Due sono le principali categorie:
— plotter;
— hard-copy.
Dei plotter abbiamo già diffusamente parlato in precedenza: vedremo pertanto ora cos'è un hard-copy.
Le unità di hard-copy hanno, contrariamente ai plotter, un'unica tipologia di rappresentazione e un unico formato, l'UNI A4. Sono paragonabili a copiatrici che, anziché copiare da un foglio di carta, copiano l'immagine dallo schermo.
Le differenze fra plotter e hard-copy sono essenzialmente due, oltre alla già accennata limitazione del formato:
— il plotter funziona sotto il diretto controllo del software grafico, mentre l'hard-copy è comandato dall'hardware del display grafico di cui costituisce un utile accessorio;
— l'hard-copy produce un output di dimensioni e risoluzione uguale o inferiore a quella del display a cui è collegato, mentre il plotter non ha problemi di dimensioni e spesso la qualità dell'immagine è superiore a quella dello schermo.
Le tecnologie utilizzate nelle hard-copy sono sostanzialmente quelle disponibili per i plotter: elettrostatica, a getto d'inchiostro (*ink-jet*, in inglese), a impatto.

Queste quattro immagini sono elaborazioni sul tema della spirale logaritmica, una figura geometrica che non è raro incontrare nel mondo naturale (le prime due immagini della pagina a sinistra riproducono la forma della conchiglia a spirale delle ammoniti). La grafica al calcolatore è utile anche negli studi biologici sulle forme naturali.



AGM - ARCHIVIO EIDOS

# I SIMULATORI DI MANOVRA

EVAN AND SUTHERLAND CORPORATION

Un'applicazione tecnologicamente avanzata della computergrafica è costituita dai "simulatori di manovra". La fedeltà è tale che il pilota, pur stando davanti allo schermo, avrà l'impressione di pilotare realmente l'aereo e si comporterà come nella situazione reale.

Un grande numero di applicazioni possono beneficiare dello sviluppo della computergrafica. Una delle più significative è l'utilizzo della tecnologia informatica per l'addestramento dei piloti d'aereo e di nave mediante speciali sistemi detti "simulatori di manovra". Questi sono costituiti da apparecchiature molto complesse, dotate di una ricca componentistica elettronica e di una potenza di calcolo che consente di fornire risposte in tempo reale alle richieste dell'operatore; ossia con una velocità tale da simulare perfettamente sotto l'aspetto temporale le situazioni che si presentano ai piloti. Il coinvolgimento dell'operatore è così presente nella dinamica globale che il sistema risulta essere un eccezionale banco di prova per la valutazione delle capacità dell'uomo, che è impegnato con i suoi riflessi, con la sua capacità di deduzione e con la sua abilità nel perseguire un obiettivo.
Sistemi con velocità di risposta di questo tipo vengono detti "sistemi in tempo reale" (real-time). I simulatori di manovra sono quindi sistemi in tempo reale e inducono nell'uomo lo stesso comportamento che avrebbe interagendo con i sistemi che vengono simulati.
I "simulatori" possono essere descritti scomponendo la loro descrizione in "parte interna" e "parte esterna". Per "parte interna", si intende l'ambiente nel quale opera chi fa la manovra; per "parte esterna", invece, una struttura basata anch'essa su elaboratori elettronici, che controllano un'apparecchiatura costituita, per esempio, dalla carlinga di un aereo vero, sostenuta da braccia oleopneumatiche, le quali, mediante un gioco di snodi, possono conferirle un'ampia scelta di assetti. Inoltre, tutte le vetrature della carlinga stessa sono coperte da sistemi di proiezione visiva che simulano, per chi sta all'interno, il paesaggio che si vedrebbe durante un certo volo. All'interno, la cabina di pilotaggio è assolutamente vera e durante la simulazione del volo anche gli strumenti di bordo tipici di un velivolo rispondono con valori identici a quelli di un volo vero. È così possibile simulare in toto, per esempio, l'atterraggio in un ben definito aeroporto, con tutti i rumori che si sentirebbero, con le comunicazioni radio, le vibrazioni e gli assetti che l'aereo verrebbe ad assumere in seguito alle manovre eseguite dal pilota.
L'obiettivo a cui tendono i simulatori di manovra è quello di raggiungere un punto tale di realismo da poter trarre in inganno anche un esperto pilota. Ossia, per esempio, si dovrebbe poter prendere un pilota di aereo già esperto sia di volo reale che di volo simulato e portarlo nel sonno in uno dei due sistemi (aereo vero e simulatore di volo): l'uomo non deve essere in grado di distinguere se si tratta di una simulazione o di un volo vero. Allo stato attuale della tecnologia non si è poi tanto lontani da questo obiettivo.
I simulatori di manovra sono oggi sempre più diffusi, data la loro efficacia nell'addestramento e nel risparmio che consentono alle compagnie aeree e navali. Basti pensare infatti a quali danni economici, oltre al rischio di perdere vite umane, può causare un addestramento fatto su un aereo o su una nave vera in caso di manovra totalmente sbagliata. Anche in questo tipo di applicazione quindi viene messo l'accento sulle enormi capacità di simulazione da parte del computer. Questa caratteristica sta sempre più rivoluzionando il modo di apprendere dell'uomo, e non è escluso che in un futuro, forse non molto lontano, le nostre prime esperienze in un qualsiasi tipo di attività vengano effettuate su un simulatore elettronico.

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 ricevere maggiori informazioni, ritagliate questo coupon e speditelo a: Olivetti M10 Piano di Ivrea
Ing. C. Olivetti & C., S.p.A. - Divisione Personal Computer - Via Meravigli 12
20123 Milano

NOME COGNOME

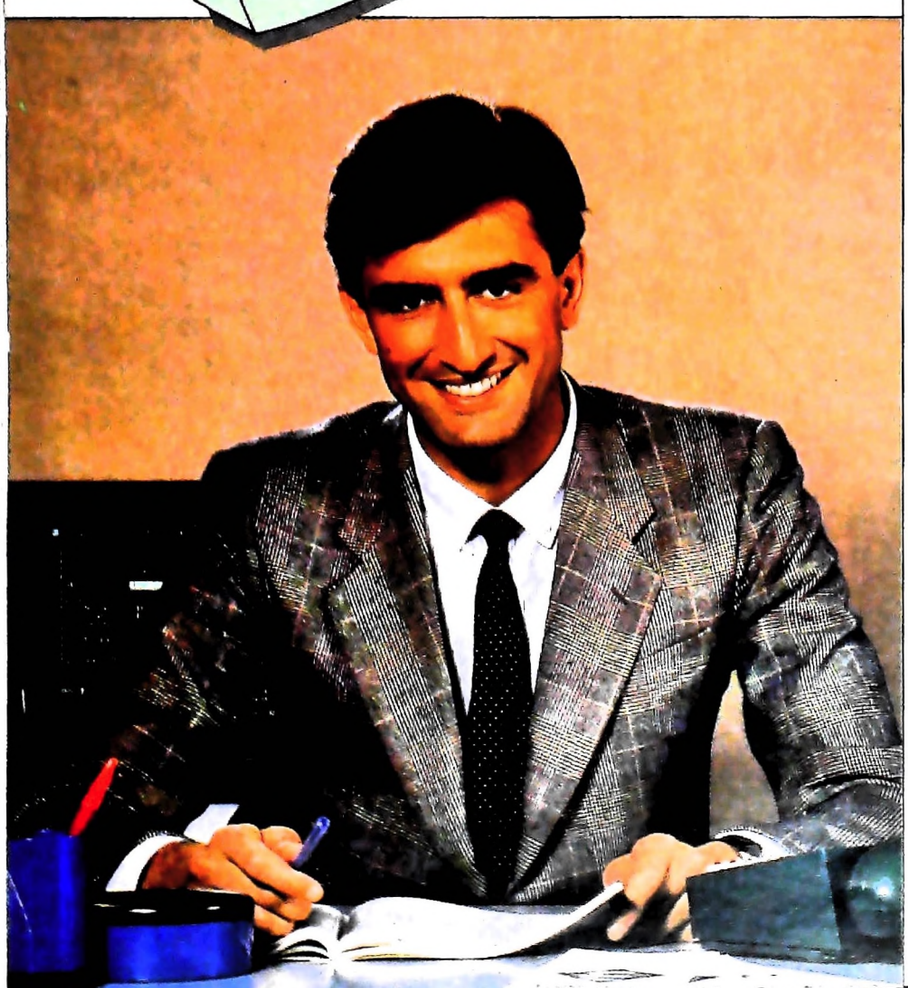
VIA

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.**