CADEL

11 CORSO PRATICO COL COMPUTER

421636

diretto da GIANNI DEGLI ANTONI

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

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

**IN OMAGGIO
IL QUARTO POSTER
"LA STORIA
DELL'INFORMATICA"**

FABBRI
EDITORI

Nel 1980 la tradizione e l'esperienza della Fabbri Editori
nel campo dell'informazione e della divulgazione culturale si sono unite
all'alto standard qualitativo di nomi come Bompiani, Sonzogno, Etas Libri.
Si è così formato un gruppo leader nel mondo italiano e internazionale
del libro. Il suo nome è

GRUPPO EDITORIALE FABBRI, BOMPIANI, SONZOGNO, ETAS S.p.A.

FABBRI. Una casa editrice specializzata in libri d'arte, manualistica d'alto livello, libri scolastici e parascolastici. La più qualificata nel rivolgersi direttamente alle famiglie con opere enciclopediche, di alto contenuto e valore, che coprono una vasta gamma di interessi. Famosa per la sua capacità divulgativa, è conosciuta e apprezzata soprattutto in edicola dove è leader indiscussa nella vendita dei fascicoli: cultura per tutti, settimana per settimana.

SONZOGNO. La grande narrativa di puro intrattenimento: i famosissimi romanzi di Liala, la scrittrice che ha fatto sognare milioni di lettrici; i bestseller di autori come Robbins, Collins; i libri da cui sono stati tratti film popolari come "Kramer contro Kramer". E altre appassionanti storie che richiamano migliaia di lettori.

BOMPIANI. La migliore narrativa in tutte le librerie: nomi come quelli di Alberto Moravia e Umberto Eco, bestseller come "Uccelli di Rovo", pubblicazioni come il "Dizionario delle opere e dei personaggi di tutti i tempi e di tutte le letterature" sono un'esclusiva Bompiani, la casa editrice che vanta il fiore della cultura mondiale di ieri e di oggi. Bompiani: da quest'anno anche in edicola con due nuove, prestigiose iniziative editoriali a fascicoli.

ETAS. Quanto di meglio si possa trovare per l'approfondimento di tematiche inerenti il mondo del lavoro, dall'economia all'informatica: marketing, pubblicità, gestione aziendale, vendite. Tutti gli argomenti più attuali in volumi spesso consigliati nelle università e nei corsi di specializzazione.

Direttore dell'opera
GIANNI DEGLI ANTONI

Comitato Scientifico
GIANNI DEGLI ANTONI
Docente di Teoria dell'Informazione, Direttore dell'Istituto di Ciberne[tica]
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 pres[so]
l'Istituto di Cibernetica dell'Università degli Studi di Milano

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

DANIELE MARINI
Ricercatore universitario presso l'Istituto di Cibernetica dell'Univers[ità]
degli Studi di Milano

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

Testi
Eidos (TIZIANO BRUGNETTI), ADRIANO DE LUCA, VIRGINIO SALA,
GOFFREDO HAUS, 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
- Milano

Direttore Editoriale
ORSOLA FENGHI

Coordinatore settore scientifico
UGO SCAIONI

Redazione
MARINA GIORGETTI
LOGICAL STUDIO COMMUNICATION

Art Director
CESARE BARONI

Impaginazione
BRUNO DE CHECCHI
PAOLA ROZZA

Programmazione Editoriale
ROSANNA ZERBARINI
GIOVANNA BREGGÉ

Segretarie di Redazione
RENATA FRIGOLI
LUCIA MONTANARI

INSIEMI E RELAZIONI

Quello di "insieme" è un concetto apparentemente banale, ma dalle estesissime applicazioni.

Un insieme è semplicemente una collezione di oggetti, concreti o astratti. Possiamo parlare dell'insieme di tutti i gatti, dell'insieme dei numeri naturali, dell'insieme di tutti i programmi in BASIC scritti dal signor Rossi, o dell'insieme costituito dalle lettere A, B, C, D. Da questa idea molto semplice i matematici hanno costruito una teoria di grande portata, che sta alla base della logica matematica moderna, e che offre un "linguaggio" molto comodo per tutti i settori della matematica. Qui non affronteremo gli aspetti più astratti della teoria degli insiemi: vedremo solo alcuni concetti di fondo, che possono essere trattati anche in modo intuitivo. Questi concetti ci serviranno in seguito per trattare l'algebra booleana, e si possono trovare frequentemente nei testi di informatica.

Un insieme è definito semplicemente dai suoi elementi, che possono essere dati "per enumerazione", cioè elencandoli uno per uno, o mediante una proprietà posseduta da tutti gli elementi dell'insieme e solo da essi. Per comodità diamo una notazione per rappresentare gli insiemi: $\{A, B, C, D\}$ è l'insieme costituito dalle lettere A, B, C e D (definizione per enumerazione); $\{x \mid x \text{ è un numero pari}\}$ è l'insieme di tutti gli oggetti x che sono numeri pari, cioè l'insieme dei numeri pari (definizione mediante proprietà).

Gli oggetti che costituiscono un insieme sono i suoi elementi e lo definiscono completamente, cioè due insiemi che contengano gli stessi elementi sono del tutto identici: l'insieme dei quadrilateri e quello dei quadrangoli coincidono, anche se sono definiti da proprietà diverse.

Un insieme A si dice sottoinsieme dell'insieme B se tutti gli elementi di A sono anche elementi di B: $A = \{1, 2\}$ è un sottoinsieme di $B = \{1, 2, 3, 4\}$.

Operazioni su insiemi

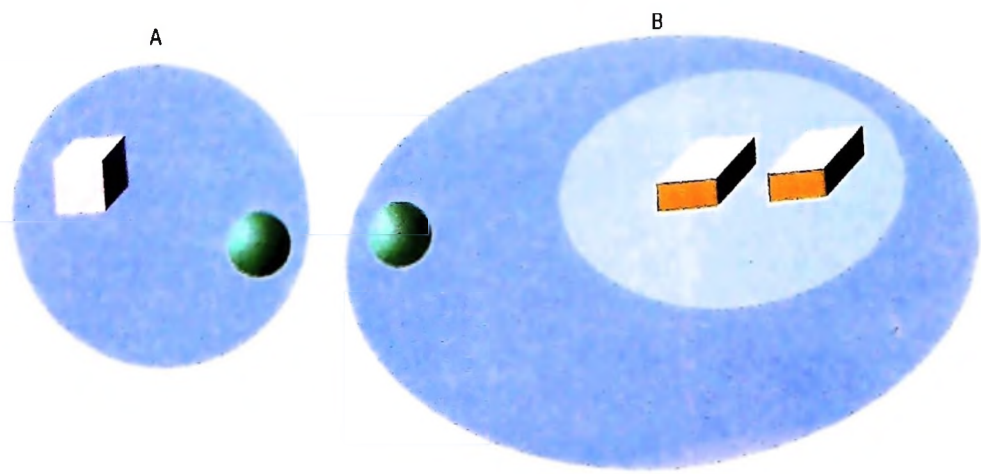
Dati due insiemi A e B vi sono alcune semplici operazioni che a partire da questi consentono la costruzione di altri insiemi.

Per cominciare possiamo definire un insieme C che sia la loro *unione* (in simboli: $C = A \cup B$), cioè l'insieme che contiene come elementi tutti gli elementi di A e tutti quelli di B. Se $A = \{1, 2, 3, 4\}$ e $B = \{10, 13\}$, la loro unione $A \cup B$ è l'insieme $\{1, 2, 3, 4, 10, 13\}$.

L'*intersezione* di due insiemi A e B (in simboli $A \cap B$) è l'insieme C i cui elementi sono gli elementi comuni di A e B. Se $A = \{1, 2, 3, 4\}$ e $B = \{x \mid x \text{ è pari}\}$, la loro intersezione è

La notazione della teoria degli insiemi

Simbolo	Significato	Esempio
$\{\ldots\}$	un insieme; ciò che è racchiuso tra parentesi sono gli elementi elencati oppure dati mediante una proprietà	$\{1, 2, 3, 4\}$ l'insieme dei primi quattro numeri naturali $\{x \mid x \text{ è pari}\}$ l'insieme degli x che sono pari
$\in$	appartenenza	$x \in A$ x è un elemento dell'insieme A
$\subseteq$	inclusione	$A \subseteq B$ A è un sottoinsieme di B (eventualmente coincidente con B)
$\subset$	inclusione propria	$A \subset B$ A è un sottoinsieme proprio di B (è escluso il caso A=B)
$\cup$	unione	$A \cup B$ l'insieme degli elementi di A e degli elementi di B
$\cap$	intersezione	$A \cap B$ l'insieme degli elementi comuni ad A e B
$-$	differenza	$A - B$ l'insieme degli elementi di A che non sono elementi di B
C	complemento	CA l'insieme degli elementi dell'universo che non appartengono ad A
$\emptyset$	insieme vuoto	
$/$	simbolo di negazione	$x \notin A$ x *non* è elemento di A $B \not\subseteq A$ B *non* è un sottoinsieme di A
$\langle\ldots\rangle$	coppia (o ennupla) ordinata	$\langle a, b\rangle$ la coppia ordinata costituita da a e b

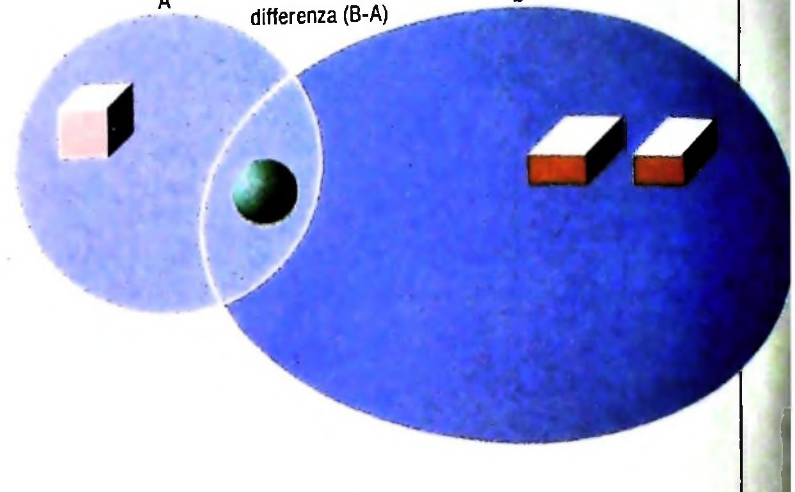


L'unione dell'insieme *A*, composto dal cubo e dalla sfera, e dell'insieme *B*, composto dalla sfera e dai due parallelepipedi, è l'insieme che contiene il cubo, la sfera e i parallelepipedi. Si noti che la sfera, l'elemento comune di *A* e *B*, si conta una volta sola. Proprio l'insieme di tutti gli elementi comuni ai due insiemi di partenza costituisce la loro intersezione; in questo caso si tratta appunto della sfera. La differenza tra due insiemi (il disegno raffigura la differenza *B* – *A*) è l'insieme composto da tutti gli elementi del primo che non appartengono al secondo.

l'insieme $C = \{2, 4\}$. Se A e B fossero gli insiemi dell'esempio precedente, la loro intersezione non conterrebbe alcun elemento, non avendo i due insiemi elementi comuni: i due insiemi si dicono *disgiunti*. Anche in questo caso, tuttavia, diciamo che l'intersezione ci dà un insieme: l'insieme vuoto, che si indica con $\varnothing$. Si noti che l'intersezione di A e B è un sottoinsieme sia di A sia di B; per coerenza, dobbiamo supporre che l'insieme vuoto sia un sottoinsieme di tutti gli insiemi.
Possiamo poi definire l'insieme *differenza* fra gli insiemi A e B, in simboli $A - B$: l'insieme di tutti gli elementi di A che non sono anche elementi di B. Un esempio: se $A = \{1, 2, 3, 4\}$ e $B = \{2, 4, 8, 16\}$, $A - B$ è $\{1, 3\}$.
L'insieme delle parti di un insieme A è l'insieme che ha come elementi tutti i sottoinsiemi di A, e si indica con "$\wp$" (A). Se A è, al solito, $\{1, 2, 3, 4\}$, il suo insieme delle parti sarà $\{\varnothing, \{1\}, \{2\}, \{3\}, \{4\}, \{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}, \{1,2,3\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}\}$. Si ricordi che ogni insieme è sottoinsieme di se stesso e che l'insieme vuoto è sottoinsieme di qualunque insieme.

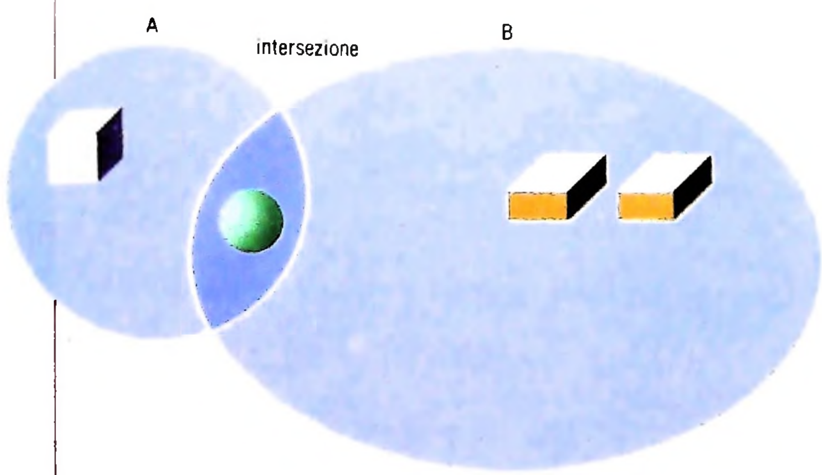
Universo e complemento

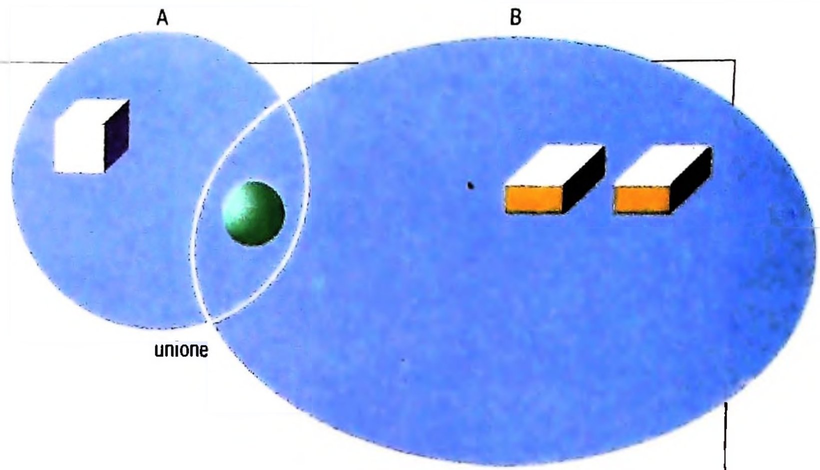
Normalmente si considerano insiemi all'interno di un ben definito "universo di discorso"; si parla, per esempio, di matematica e l'universo di discorso può essere rappresentato dai numeri naturali positivi, zero incluso (chiamiamo N questo insieme). (Non conviene, mantenendosi sul piano intuitivo, fare riferimento a un universo "totale": si corre il rischio di finire in situazioni paradossali, come è successo storicamente ai "padri fondatori" della teoria degli insiemi. Per evitare i paradossi è necessario far ricorso a teorie formali, che qui sono fuori dal nostro interesse, o mantenersi all'interno di totalità "piccole", per conservare un'impostazione intuitiva.)
È comodo, dato un universo di discorso o "insieme totale", introdurre un'operazione di complemento: dato un insieme A, il complemento CA, è l'insieme di tutti gli elementi dell'insieme totale che non appartengono ad A. Così, se $A = \{0,1,2,3,4\}$, e l'universo di discorso è costituito da N, $CA = \{x \mid x > 4\}$.

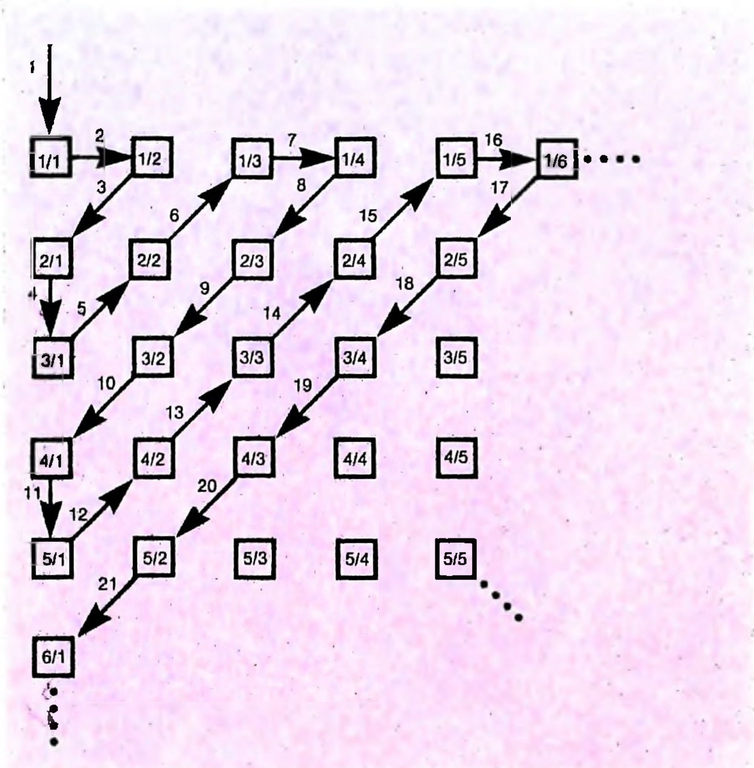


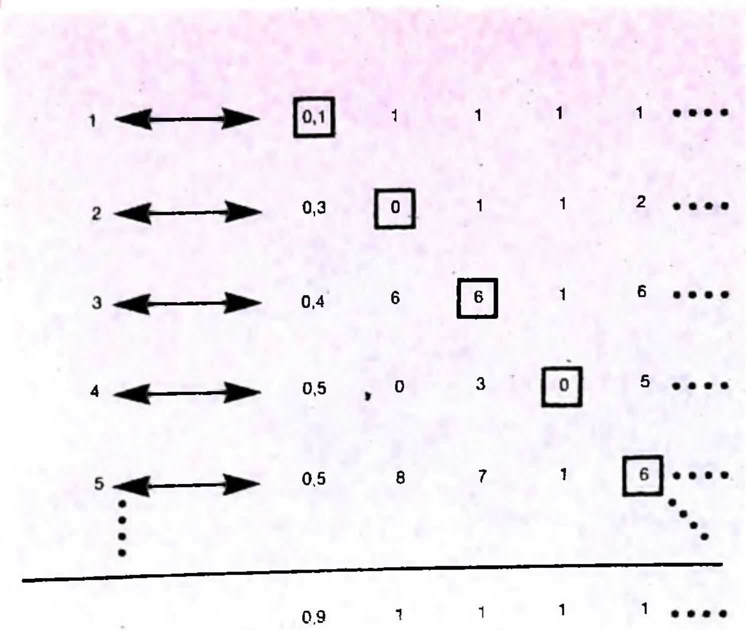
Insiemi ordinati

Quando si parla di insiemi, si intende generalmente parlare di collezioni di elementi qualunque, collezioni non strutturate internamente; spesso però è utile parlare di collezioni "ordinate", all'interno delle quali è definito cioè un ordine degli elementi. Un insieme ordinato viene rappresentato in genere con parentesi angolari, anziché graffe: $\langle 1, 2\rangle$ è l'insieme ordinato che ha due elementi, il primo dei quali è 1, il secondo 2; $\langle 1, 2\rangle$ è diverso da $\langle 2, 1\rangle$, anche se ambedue contengono i due soli elementi 1 e 2. Si parla spesso, in un caso del genere, di "coppia ordinata" e, analogamente, di tripla, quadrupla ecc., ordinata per insiemi ordinati con tre, quattro ecc., elementi.
Un insieme si dice ordinato quando sui suoi elementi è definita una relazione d'ordine, una relazione che fissa la priorità di un elemento rispetto a un altro. La più nota relazione d'ordine è quella che sussiste per la successione dei naturali: 1 precede 2, 2 precede 3, 3 precede 4, e via di seguito. Questa è una relazione d'ordine totale: dati due naturali qualunque a e b, o a precede b o b precede a. Possono darsi però anche relazioni d'ordine parziali, quando non tutti gli elementi sono confrontabili fra loro. Pensate per esempio a un albero genealogico: il bisnonno precede il nonno e il nonno precede









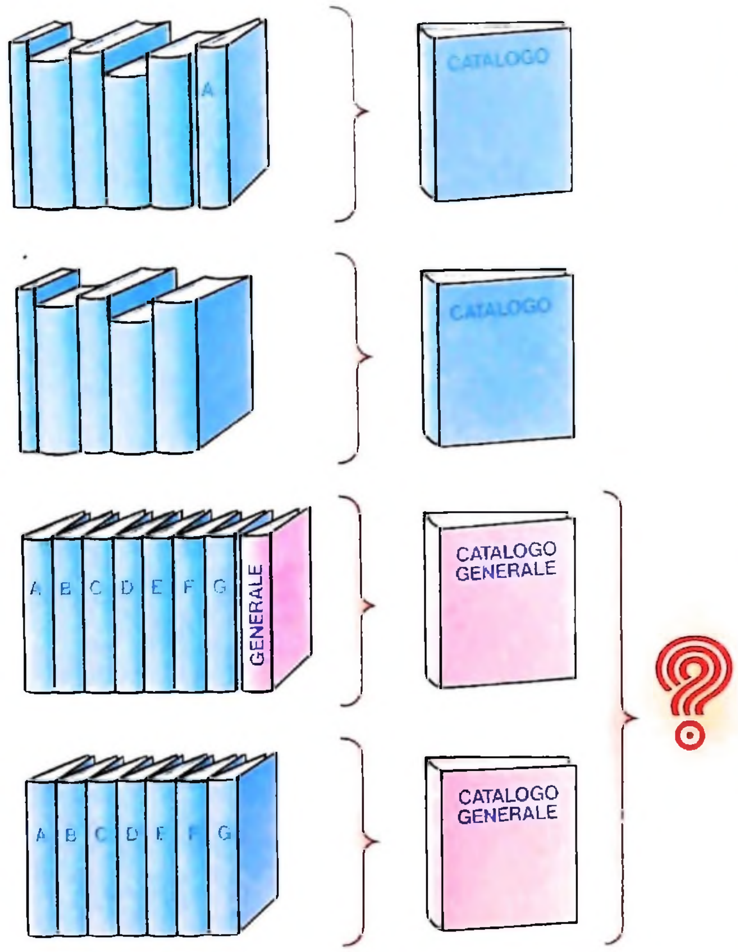
Nonostante le apparenze, è possibile mettere in ordine i numeri razionali, cioè le frazioni, assegnando a ciascuno un numero intero in maniera unica e senza equivoci. Il trucco è in realtà abbastanza semplice. Basta disporli nella tabella infinita, qui a lato, in cui lungo le righe aumenta il numeratore e rimane costante il denominatore e lungo le colonne viceversa. Per contarli, poi, basta seguire il percorso indicato dalla freccia. In questo modo si può raggiungere qualunque frazione muovendosi di un certo numero (finito) di passi lungo la tabella, e quindi assegnarle un posto nell'ordinamento. Non tutti gli insiemi infiniti possono essere posti in corrispondenza biunivoca: alcuni sono "più infiniti" di altri.
Così è, per esempio, per l'insieme dei numeri reali e quello degli interi. La dimostrazione di questo è rappresentata nel disegno in basso; supponiamo di aver posto in corrispondenza i reali compresi fra 0 e 1 con gli interi: si può costruire un nuovo reale non compreso nella corrispondenza e diverso da tutti gli altri, che differisce dal primo per la prima cifra decimale, dal secondo per la seconda cifra decimale, e via di seguito.

il padre; il padre precede i figli, ma i figli non sono confrontabili fra loro.
Una relazione d'ordine parziale può sempre essere visualizzata come un albero: quando due elementi giacciono su ramificazioni diverse non sono confrontabili (in quella relazione d'ordine).
Un'altra importante relazione d'ordine è quella di inclusione fra insiemi: quando un insieme è contenuto in un altro (è un suo sottoinsieme) "lo precede". La relazione non è totale, perché due insiemi possono essere disgiunti, o avere elementi non in comune.

Insiemi e relazioni

La relazione d'ordine è un particolare tipo di relazione definita su un insieme, ma possiamo considerare a sua volta ogni relazione come un insieme, più precisamente come l'insieme delle coppie ordinate di elementi che stanno in quella relazione.
Prendiamo per esempio la relazione "essere divisore di": due numeri a e b stanno in questa relazione quando a è un divisore di b. Possiamo considerarla come l'insieme di tutte le cop-

Un paradosso sull'insieme di tutti gli insiemi



Trattando insiemi non inseriti in un "universo del discorso" ben limitato senza usare assiomi sofisticati si incontrano paradossi insuperabili. Un esempio particolarmente semplice si può trovare considerando le biblioteche. Il catalogo di *tutti i libri* di una biblioteca può essere un volume molto grosso. Bisogna inserirlo tra le sue stesse voci? Supponiamo che in alcune biblioteche locali si decida di no, e in altre di sì. Ora però il direttore della biblioteca centrale decide di fare un catalogo di tutti quei cataloghi che *non* citano se stessi. Questo grande catalogo generale dovrà citare se stesso o no? È facile vedere che nessuna delle due possibilità è soddisfacente, cioè che il catalogo deve contemporaneamente citarsi e non citarsi.

pie ordinate di numeri naturali ‹a,b› tali che a divide b. In simboli, R = {‹a, b› | a è un divisore di b}.
Questa soluzione ammette un viceversa: qualunque insieme di coppie ordinate può essere considerato una relazione (anche se non abbiamo un nome convenzionale da dare alla relazione, come nel caso dell'essere divisore). Ammette anche una generalizzazione: le relazioni non sono solo binarie, ma possono coinvolgere anche più elementi: per esempio, "essere punto medio fra" è una relazione ternaria. Una relazione n-aria è rappresentabile come un insieme di ennuple ordinate. Per esempio: {‹a, b, c› | a è punto medio fra b e c} è l'insieme che corrisponde alla citata relazione ternaria. Si dice anche che l'insieme è l'*estensione* o il *grafo* della relazione.

Corrispondenze, relazioni e funzioni

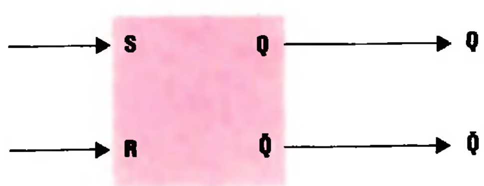
Dati due insiemi A e B, si può sempre costruire una corrispondenza fra elementi dell'uno ed elementi dell'altro. Il modo più semplice di visualizzare la cosa è rappresentare i due insiemi con i diagrammi di Venn, come si può vedere negli schemi delle pagine precedenti, e evidenziare la corrispondenza con archi che uniscono i punti che rappresentano gli elementi corrispondenti dei due insiemi.
L'idea è in effetti più semplice di quel che appaia esprimendola con delle parole.
Qui c'è un insieme di caramelle, là un insieme di bambini. Si può creare una corrispondenza fra i due insiemi dando delle caramelle a dei bambini. Come matematici, ci interessa vedere anche quali caratteristiche ha la corrispondenza. Può darsi che non tutti i bambini ricevano una caramella, e che quindi la corrispondenza sia solo con un sottoinsieme dell'insieme dei bambini; può darsi però che ogni bambino riceva almeno una caramella, e in questo caso si dice che la corrispondenza è *suriettiva*. Il caso più comune sarà che a ogni caramella corrisponda un solo bambino: la corrispondenza è allora *univoca*. D'altra parte, qualche bambino potrebbe ricevere più di una caramella; se ciò non si verifica, la corrispondenza si dice *iniettiva*. Infine, se ogni caramella va a un bambino solo e ogni bambino riceve una caramella sola, diciamo che la corrispondenza è *biunivoca*. Introduciamo un po' di terminologia matematica. Se la corrispondenza f è dall'insieme A all'insieme B, A viene detto il *dominio* di f, B il suo *codominio*. Se a è un elemento di A e b è l'elemento di B che gli corrisponde attraverso la f, scriviamo $b = f(a)$ e diciamo che b è l'immagine di a nella f. Quando tutti gli elementi di B sono immagine di qualche elemento di A, f è suriettiva. Se inoltre ogni elemento di A ha una sola immagine in B, f è iniettiva.
Se, oltre a questo, ogni elemento di B è immagine di un solo elemento di A, f è biiettiva.
Si definisce *prodotto cartesiano* di due insiemi A e B (che si indica $A \times B$) l'insieme di tutte le coppie ordinate ‹a, b›, in cui a è un elemento di A e b un elemento di B. Una corrispondenza tra A e B allora può essere considerata un sottoinsieme del prodotto cartesiano di A e B. (A e B possono anche non essere diversi: $A \times A$ è il prodotto cartesiano di A con se stesso, cioè l'insieme di tutte le coppie formate da elementi di A, che spesso si indica A^2).
Quelle che solitamente chiamiamo *funzioni* non sono altro che corrispondenze univoche in questo senso insiemistico. La funzione sen, per esempio, è una corrispondenza fra l'insieme dei numeri reali e l'insieme dei reali compresi fra 0 e 1 (estremi inclusi). Possiamo considerarla, insiemisticamente, come un insieme di coppie ordinate ‹a, b›, in cui a è un numero reale qualunque, e b è sen a (reale compreso fra 0 e 1). Non è evidentemente una funzione biunivoca.
Quando una funzione è biunivoca, si può definire la sua inversa. Se f è quella corrispondenza fra l'insieme dei naturali e l'insieme dei numeri pari che a ogni naturale fa corrispondere il suo doppio, l'inversa di f (f^{-1}) è quella corrispondenza fra l'insieme dei numeri pari e l'insieme dei numeri naturali che a ogni pari fa corrispondere la sua metà.

IL FLIP-FLOP E I CIRCUITI SEQUENZIALI

Il circuito bistabile su cui si basa la memoria e le relative applicazioni.

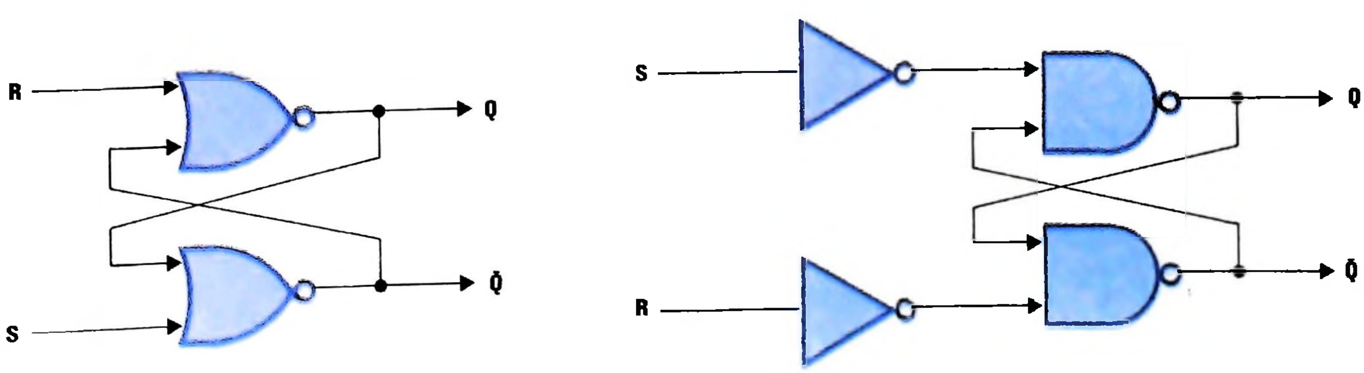
I flip-flop sono circuiti digitali molto importanti, che svolgono un ruolo decisivo nell'integrazione dei sistemi. Vediamo ora come sono fatti e come funzionano.
Il primo flip-flop preso in considerazione è quello SR (Set-Reset), mostrato nella figura 1. In esso sono possibili due stati, indicati con i simboli Q e Q̄, dove Q̄ è considerata l'uscita negativa, o ATTIVA NEGATIVA di Q. Esso ha due segnali d'entrata, la linea SET(S) e la linea RESET(R). Con i valori di ingresso S=1 e R=0, all'uscita avremo Q=1 e Q̄=0, vale a dire le due uscite sono nel loro stato ATTIVO, mentre con S=0 e R=1 le due uscite sono Q=0 e Q̄=1, cioè non ATTIVO.
Quando si ha S=R=0, il flip-flop non cambia stato, mentre se si ha S=R=1 l'uscita è indefinita, per cui questa condizione non è permessa. La figura 2 mostra come si realizza un flip-flop tipo SR.
Prima di passare all'analisi degli altri tipi di flip-flop, sarà meglio imparare a interpretare le Tabelle degli Stati, che sono presentate insieme agli schemi dei flip-flop.
In ogni Tabella compaiono sempre tre parametri: le Entrate, lo Stato presente, e lo Stato prossimo. Prendiamo, per esempio, la prima riga della Tabella degli Stati del flip-flop RS, che abbiamo appena visto; il suo significato è che quando, al tempo = t, abbiamo le due entrate S(t) e R(t) e lo Sta-

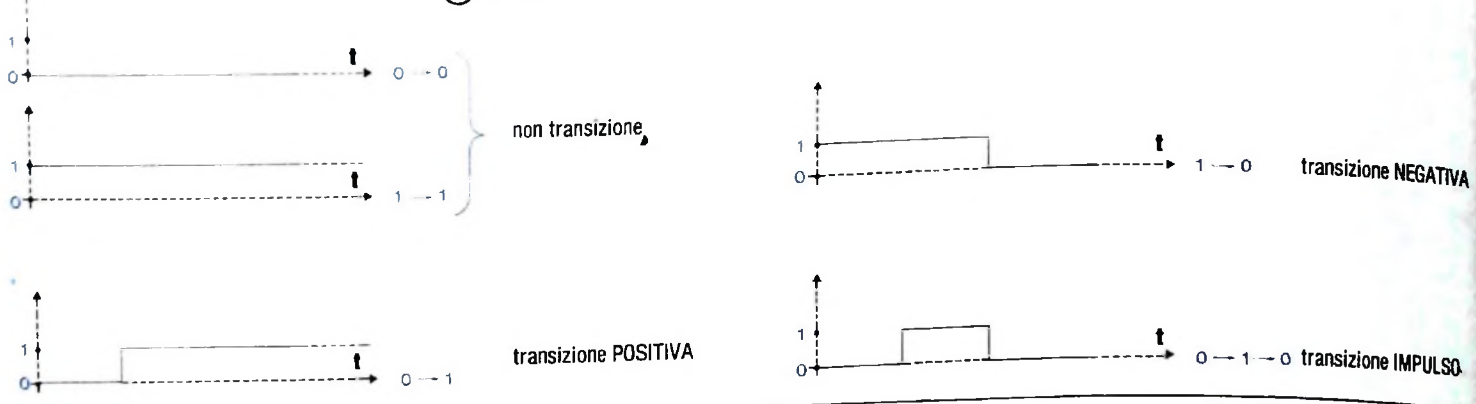
(1) Il flip-flop SR (Set-Reset): a sinistra il suo simbolo, a destra la Tabella degli Stati.



Input		Stato presente	Prossimo stato	
S(t)	R(t)	Q(t)	$Q(t+\varepsilon)$	
0	0	0	0	Nessun cambiamento
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	d	Non ammesso
1	1	1	d	

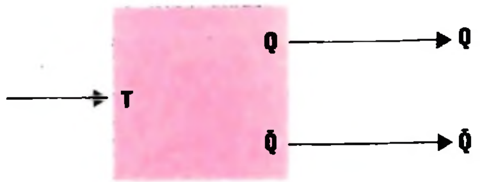
(2) Due realizzazioni pratiche del flip-flop SR.





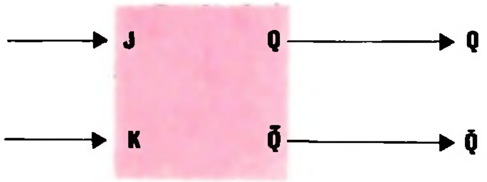
(3) Rappresentazione grafica degli Stati di transizione.

(4) Il flip-flop T: a sinistra il suo simbolo, a destra la Tabella degli Stati.



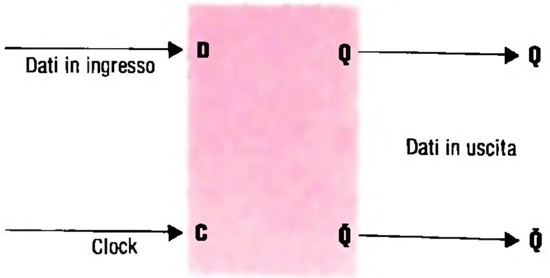
Input	Stato presente	Prossimo stato	
T(t)	Q(t)	Q(t+ε)	
0→ 0,1→ 0,1→ 1	0	0	Nessun cambiamento
0→ 0,1→ 0,1→ 1	1	1	
0→ 1	0	1	Complemento
0→ 1	1	0	

(5) Il flip-flop JK: a sinistra il suo simbolo, a destra la Tabella degli Stati.



Input		Stato presente	Prossimo stato	
J(t)	K(t)	Q(t)	Q(t+ε)	
0	0	0	0	Nessun cambiamento
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Complemento
1	1	1	0	

(6) Il flip-flop D: a sinistra il suo simbolo, a destra la Tabella degli Stati.



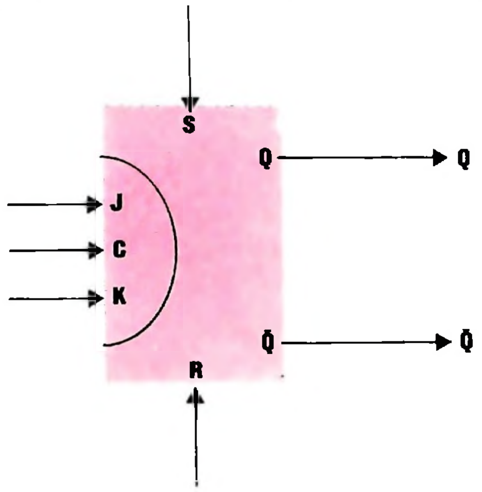
Input		Stato presente	Prossimo stato	
D(t)	C(t)	Q(t)	Q(t+ε)	
0	0→0,1→1,1→0	0	0	Nessun cambiamento
0	0→0,1→1,1→0	1	1	
0	0→1	0	0	Dati sincronizzati in ingresso
0	0→1	1	0	
1	0→0,1 1,1→0	0	0	Nessun cambiamento
1	0→0,1 1,1→0	1	1	
1	0→1	0	1	Dati sincronizzati in ingresso
1	0→1	1	1	

to presente della uscita Q uguali a zero, allora, al tempo t+ε, cioè immediatamente dopo lo stabilizzarsi dei segnali, lo stato di Q diventa 0, cioè nel nostro caso non cambia (cambia invece nella quarta e quinta riga, rispettando le regole di funzionamento che abbiamo visto sopra). Altra configurazione importante in queste tabelle è quella che compare in alcuni tipi di flip-flop, che vedremo in seguito (figura 3) nelle Entrate 0→0, 1→0, 0→1, e 1→1, dove le frecce vogliono dire "transizione": va però sottolineato che le configurazioni 0→0 e 1→1 significano che non c'è stata transizione, mentre 1→0 e 0→1 indicano che sono avvenute transizioni, NEGATIVE e POSITIVE rispettivamente; per ultimo, la configurazione 0→1→0 indica una transizione prima POSITIVA, e poi, dopo un certo tempo, NEGATIVA, comunemente chiamata transizione IMPULSO.

Il flip-flop T (Trigger, cioè "grilletto") è un flip-flop che cambia stato di uscita ad ogni segnale che arriva all'entrata T (figura 4). Questa entrata è legata al segnale di sincronizzazione dell'orologio. I circuiti visti finora avevano uscite che dipendevano dal valore dell'entrata in quel momento, men-

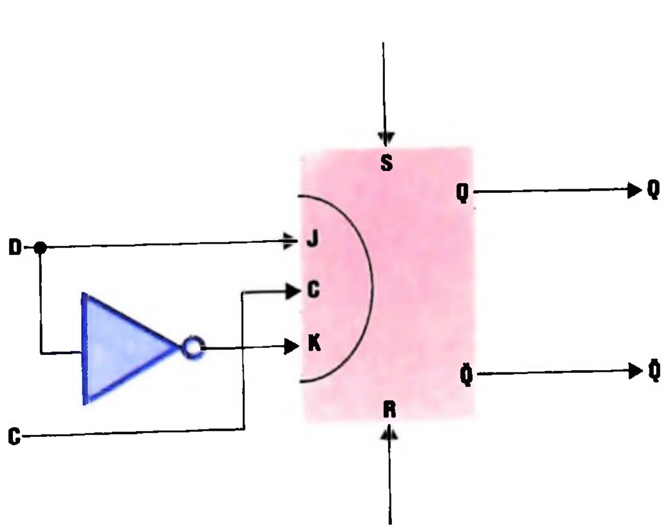
tre con il flip-flop T l'uscita dipende non solo dallo stato del circuito prima dell'arrivo del clock (segnale di temporizzazione) ma anche dall'arrivo del clock stesso, che segna esattamente l'istante in cui il circuito deve cambiare stato. Questo tipo di circuito è chiamato "sincrono", mentre i circuiti logici tipo AND, OR ecc. vengono definiti come "asincroni".
Il flip-flop JK (figura 5) è in realtà la combinazione dei due tipi di flip-flop precedenti, cioè il Set-Reset e il Trigger. Il flip-flop JK si comporta come quello SR, salvo nel caso J=K =1; in questo caso esso si comporta come un flip-flop Trigger, per cui cambia stato.
Vediamo ora il flip-flop tipo D (figura 6). Questo flip-flop, molto importante per le sue caratteristiche, è considerato l'elemento di memoria per eccellenza. Infatti D sta per Delay (ritardo). Esso presenta due entrate D e C (Clock): lo stato presente nell'entrata D passa all'uscita Q quando all'entrata C c'è uno stato di transizione — intendendo per transizione il passaggio da uno stato 0 a uno stato 1 e viceversa.
Infine vediamo una variante del flip-flop JK che è il flip-flop CLOCKED (figura 7).
Questo è un flip-flop completamente generale, che può funzionare sia come tipo T, tipo D e tipo SR; tutto dipende da come viene collegato per farlo operare secondo uno di questi modi come possiamo vedere nella figura 8.

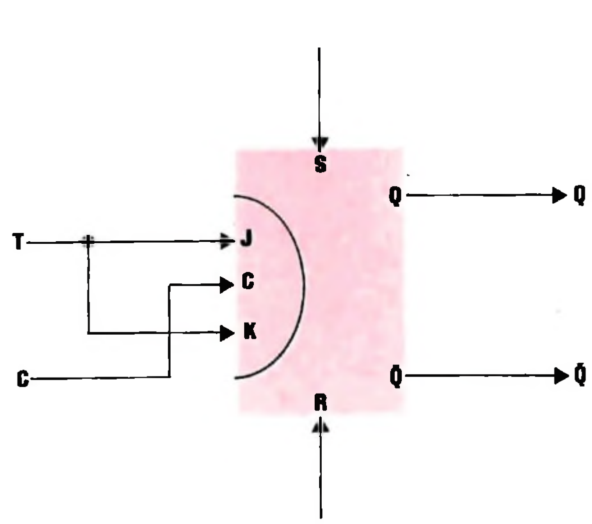
7 Il flip-flop JK CLOCKED: a sinistra, il suo simbolo, a destra la Tabella degli Stati per operazioni sincrone (S=R=0), al centro la Tabella degli Stati per operazioni statiche.



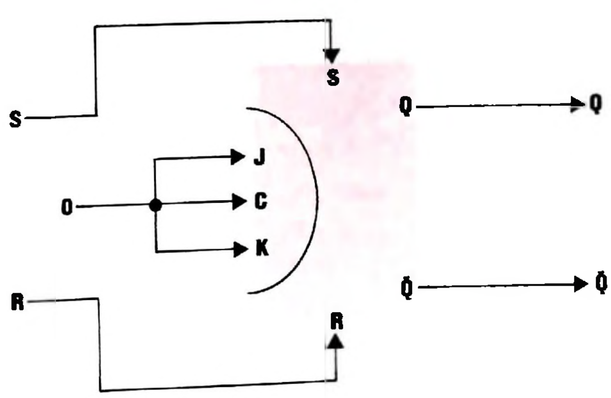
S(t)	R(t)	Q(t)	Q(t+ε)
0 0	0 0	0 1	? Operazione sincrona ? vedi la Tabella a lato
0 0	1 1	0 1	0 Reset 0
1 1	0 0	0 1	1 Set 1
1 1	1 1	0 1	d Non d ammesso

Input		Clock	Stato presente	Prossimo stato
J(t)	K(t)	C(t)	Q(t)	Q(t+ε)
0 0	0 0	0→1→0→ 0→1→0→	0 1	0 Nessun 1 cambiamento
0 0	1 1	0→1→0→ 0→1→0→	0 1	0 Reset 0
1 1	0 0	0→1→0→ 0→1→0→	0 1	1 Set 1
1 1	1 1	0→1→0→ 0→1→0→	0 1	1 Complemento 0





8 Usi del flip-flop JK CLOCKED: qui sopra come flip-flop D, sopra a destra come flip-flop T, a lato come flip-flop SR.



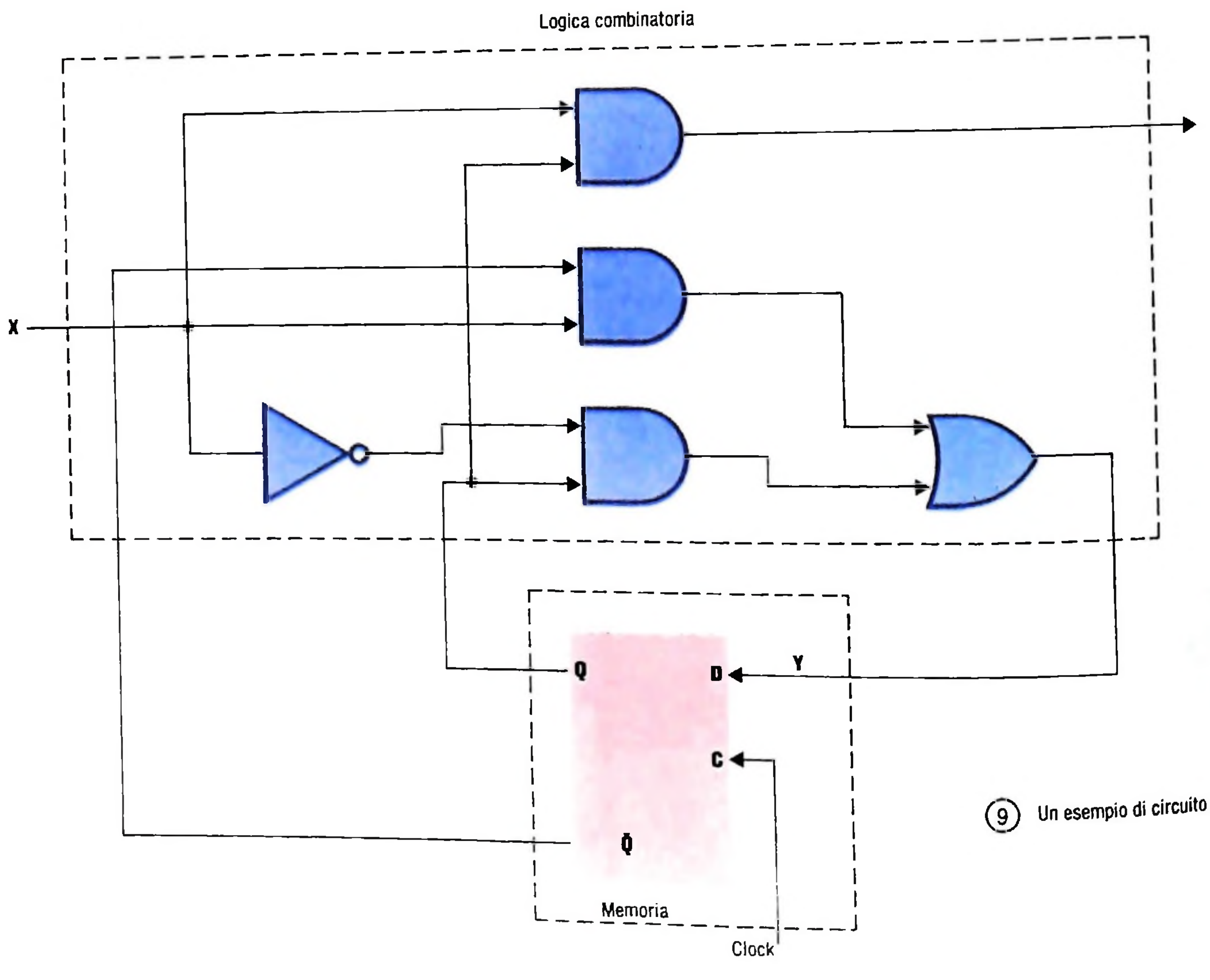
Circuiti sequenziali

Dopo la rapida carrellata sui principali circuiti digitali esistenti, prendiamo ora in considerazione due grandi famiglie di circuiti digitali complessi, cioè i circuiti combinatori e i circuiti sequenziali. Cercheremo di spiegare il loro funzionamento descrivendo i circuiti sequenziali, dato che questi ultimi includono anche i primi.

Il concetto di sequenzialità non è, comunque, ristretto ai sistemi digitali. Per illustrare questo concetto, consideriamo le operazioni di un ascensore moderno in un grattacielo di 50 piani. In questo caso, l'ascensore si comporta come un sistema sequenziale, poiché la sua azione è determinata dalla presenza, o meno, del segnale di chiamata a piani, da un comando datogli dall'interno della cabina e dalla sua posizione in uno dei 50 piani dell'edificio. L'ascensore deve, quindi, prima di tutto "ricordare" la sua posizione presente, in modo da determinare, analizzando anche gli altri parametri, esattamente qual è il prossimo piano su cui fermarsi. Prendendo come esempio l'ascensore, possiamo definire le varie fasi e i vari elementi presi in considerazione da un sistema sequenziale prima di determinare la sua azione o comunque dare un risultato. Va definito lo "Stato presente" (Present-State) dell'ascensore, come indicazione sia del piano presente che di quello precedente, allo scopo di determinare esattamente se l'ascensore stava salendo o stava scendendo.

Poi, va definito lo "Stato seguente" (Next-State): la prossima fermata dell'ascensore è determinata dal valore dello "Stato presente" e dal valore del "comando" (INPUT) che può venire sia da uno qualsiasi dei piani come pure dall'interno stesso dell'ascensore. Naturalmente, l'ascensore si troverà in uno "Stato di transizione" (Transition-State) quando si sta muovendo dal piano dove è partito al piano dove si fermerà. I concetti di "Stato presente", "Stato seguente", "Comando" e "Stato di transizione" sono fondamentali nello studio dei circuiti logici sequenziali. Ricapitolando, se ne può dedurre che per poter determinare lo Stato seguente di un sistema sequenziale si ha bisogno anzitutto di circuiti capaci di ricordare il passato e il presente, cioè di memorie. Queste si ottengono con circuiti flip-flop di tipo D, visti in precedenza, o con altri circuiti; per poter determinare l'uscita esatta in concordanza con le condizioni imposte più gli stati passati si ha bisogno di circuiti combinatori, cioè circuiti tipo AND, OR ecc.

Il modello classico di un circuito sequenziale è rappresentato nella figura 9. Un'ultima nota: se osserviamo attentamente il modello del circuito sequenziale vediamo una retroazione (*feed-back*) dell'uscita verso l'entrata. La prima domanda che sorge spontanea è se nel circuito sequenziale non si verifica l'innesco di oscillazioni che naturalmente distruggerebbero completamente la sequenza dell'informazione. Questa condizione non è possibile per il semplice motivo che in nessun momento l'entrata è in collegamento diretto con l'uscita. Come si è già visto nel flip-flop di tipo D, il segnale in entrata passa all'uscita solo nella transizione di clock, cioè in un tempo brevissimo che rende impossibile l'innesco delle oscillazioni.



(9) Un esempio di circuito sequenziale.

Lezione 10

Manipolare tabelle

Completata questa decima lezione del Corso di Programmazione e BASIC, siete in grado di eseguire gli esercizi SLOTT.DO SLOPT.BA contenuti nella cassetta "11 Esercizi di Programmazione"
I titoli seguiti dal suffisso DO corrispondono a testi, quelli seguita da BA a programmi in BASIC.
Caricateli secondo le modalità che avete appreso.

Supponiamo di voler risolvere il seguente problema: calcolare la media delle stature degli allievi di una classe e indicare quanti di essi sono più alti della media.
Abbiamo già visto altrove come calcolare la media di un certo insieme di valori, mediante un'opportuna iterazione; qui però il problema è radicalmente più complesso, in quanto si tratta di un procedimento che richiede due "passate" sui dati di partenza:

- una prima passata ha lo scopo di scandire tutti i valori per totalizzarli e calcolarne la media;
- una seconda passata ha lo scopo di scandire di nuovo tutti i valori per confrontarli con la media e quindi contare quanti di essi hanno un valore superiore.

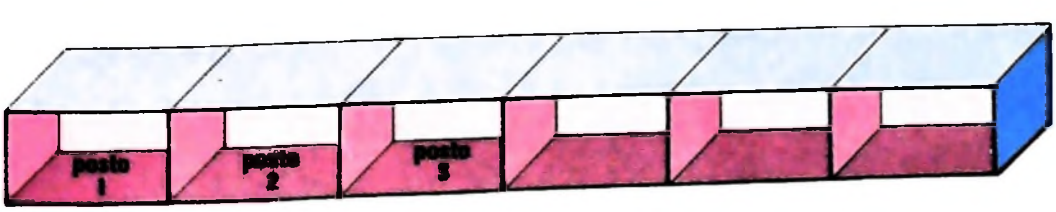
Evidentemente, il problema può essere risolto in due modi:

- fornire i valori da elaborare due volte per le due passate;
- conservare all'interno del programma i valori, dopo averli letti la prima volta, per poterli scandire la seconda.

La prima soluzione è evidentemente molto scomoda, o addirittura poco praticabile, a causa non solo degli aspetti di utenza, ma anche per la possibilità di errori nell'introduzione dei dati. La seconda, d'altra parte, non è semplice, allo stato attuale delle nostre conoscenze. Infatti, si tratterebbe di costruire un programma BASIC del seguente tipo:

```
10 INPUT A,B,C,D,E,F,G,H,I,L,M,O,P...
20 LET Z=(A+B+C+D+E+...)/N
30 LET C=0
40 IF A>Z THEN LET C=C+1
50 IF B>Z THEN LET C=C+1
60 IF C>Z THEN ...
```

dove N alla riga 20 è il numero di elementi che non è solo scomodo o lungo da scrivere, ma piuttosto profondamente legato al numero di valori da elaborare, e quindi impraticabile come programma (infatti, occorrerebbe modificarlo ogni volta che varia il numero di allievi).
Ciò di cui avremmo bisogno è di un insieme di variabili dotate di un unico nome e distinguibili tra di loro per la specifica posizione in un "casellario", come nella figura sottostante:

A


In tal caso, con A indicheremmo l'intero "casellario", con A(1) la prima variabile del casellario (cioè quella che corrisponde al posto 1), con A(17) la variabile di posto 17; con A(I), supponendo che la variabile I contenga il valore 5, la variabile di A di posto 5, e ancora, con A(I+3), la variabile di posto 8.

Allora, il nostro problema potrebbe essere risolto come segue:

- leggi N

  (lettura del casellario)

- FOR I: = 1 TO N DO

  leggi un valore in A (I)

  (calcolo media)

- S:=0
- FOR I:=1 TO N DO

  S:=S+A (I)

- M:=S/N

  (conteggio > della media)

- C:=0
- FOR I:=1 TO N DO

  IF A (I)>M THEN

  C:=C+1

- visualizza M e C

Spesso si parla di tali variabili composite come di TABELLE o ARRAY e si fa riferimento alla posizione parlando dell'"indice" dell'array. Si parla anche di MATRICI (nel caso in cui gli indici siano più di uno, come vedremo).

La realizzazione di un array in basic

Per poter disporre di un array in BASIC è necessario effettuare una specifica richiesta mediante un'istruzione DIM come la seguente:

```
10 DIM A(15)
```

che ci mette a disposizione un array di 16 elementi, numerati da 0 a 15. Non è evidentemente necessario che noi usiamo tutti tali elementi: semplicemente, quella è la massima quantità di elementi usabili e, se tentiamo per esempio di accedere a un elemento di posto 16, viene segnalato un errore (per la precisione con un codice d'errore BS).
Così, se pensiamo che la nostra classe di allievi non superi mai le 35 unità potremo scrivere il seguente programma:

```
10 DIM S(34)
20 LET C=0
30 FOR I=0 TO 34
40 INPUT "INSERISCI VALORE ALTEZZA";S(I)
50 LET C=C+S(I)
60 NEXT I
70 LET M=C/35
80 PRINT "LA STATURA MEDIA E': ";M
90 REM Calcola quante altezze > della media
100 LET N=0
110 FOR I=0 TO 34
120 IF S(I)>M THEN LET N=N+1
130 NEXT I
140 PRINT N;" SONO PIU' ALTI DELLA MEDIA"
```

Si noti che, nel programma sopra riportato, la variabile contatore che controlla il numero di iterazioni nelle due strutture FOR...NEXT assume i valori corrispondenti via via ai valori dell'indice dell'array. Per tale motivo, i valori di tale contatore sono fissati tra 0 e 34.
In qualche caso, il numero di elementi dell'array necessari non è noto a priori, ma può essere conosciuto solo durante l'esecuzione. Potremmo per esempio voler generalizzare il programma sopra riportato in modo da poterlo usare in qualsiasi contesto e non solo per quella limitata fascia di casi in cui il numero di persone di cui si vuole calcolare l'altezza media è minore o uguale a 35. Vorremmo per esempio che il programma potesse essere ugualmente usato per calcolare per esempio l'altezza media dei membri di una famiglia come pure degli abitanti di una cittadina.
Potremmo allora modificare il programma nel modo seguente:

```
10 INPUT "QUANTI DATI";A
20 DIM S(A)
30 FOR I=0 TO A-1
40 INPUT "INSERISCI VALORE ALTEZZA";S(I)
50 LET C=C+S(I)
60 NEXT I
70 LET M=C/A
80 PRINT "LA STATURA MEDIA E': ";M
90 REM Calcolo altezze > media
100 N=0
110 FOR I=0 TO A-1
120 IF S(I)>M THEN N=N+1
130 NEXT I
140 PRINT N;" SONO PIU' ALTI DELLA MEDIA"
```

Si noti come sono stati espressi i valori della variabile contatore alle linee 30 e 110.
Il programma così modificato chiede che sia fornito il numero di individui di cui si vuole calcolare l'altezza media e quindi utilizza un array delle dimensioni dichiarate. L'effetto dell'istruzione DIM è proprio quello di riservare uno spazio in memoria per l'array delle dimensioni dichiarate. Tale dimensione si può esprimere sia tramite

una costante numerica (cioè il numero in chiaro) sia attraverso una variabile. Quest'ultimo caso è ammesso solo quando il BASIC è, come si dice, "interpretato". Come vedremo meglio nel seguito, infatti, i linguaggi di programmazione si dividono in due categorie rispetto alle modalità di esecuzione dei programmi: alcuni linguaggi vengono "compilati" prima di poter essere eseguiti, altri invece non richiedono la "compilazione" e possono essere direttamente eseguiti e sono quelli che, come il BASIC dell'M10, vengono "interpretati".

Cerchiamo di capire che cosa intendiamo quando parliamo di "compilare" un programma. L'elaboratore ha un proprio linguaggio, il linguaggio binario, nel quale tutti i simboli significativi sono costruiti a partire da sequenze di zeri e uni. I programmi scritti nel linguaggio di programmazione prescelto devono prima essere convertiti nel linguaggio dell'elaboratore per poter essere da questo compresi e quindi eseguiti. Questa operazione di "traduzione" viene effettuata ancora una volta da un programma specializzato, il "compilatore". Quest'ultimo provvede a generare una versione, scritta nel linguaggio dell'elaboratore, del programma costruito nel nostro linguaggio di programmazione. Così si ottengono due versioni del programma: quella che possiamo visualizzare tramite il comando LIST, che chiameremo "programma sorgente", e quella eseguibile, o "programma oggetto".

Alcuni linguaggi o addirittura alcune versioni dello stesso linguaggio consentono di evitare questo passo intermedio di generazione dell'"oggetto" e di eseguire direttamente il "sorgente": ciò è reso possibile ancora una volta dalla presenza di un programma, l'"interprete", che provvede appunto a "interpretare" le istruzioni del "sorgente" e a eseguire le operazioni opportune relative a ciascuna di esse.

Nel caso dunque di linguaggi BASIC "compilati" non è possibile esprimere le dimensioni di un array tramite una variabile: il compilatore, infatti, non potendo prevedere il valore che tale variabile assumerà, non è in grado di "tradurre" opportunamente l'istruzione DIM. Nel caso del BASIC "interpretato" il problema non sussiste, poiché l'istruzione non è "tradotta" prima di venir eseguita, ma è direttamente eseguita tramite l'interprete quando anche i valori delle variabili sono noti.

Si noti infine che l'istruzione DIM non è obbligatoria: se non viene espressa, l'interprete BASIC allocherà dieci elementi per ogni dimensione: ciò accade anche quando l'istruzione DIM venga indicata dopo che l'array è già stato usato: in tal caso, se le dimensioni dichiarate nella DIM sono inferiori a quelle effettive dell'array, sarà fornita un'indicazione di errore di tipo DD.

Cosa abbiamo imparato

In questa lezione abbiamo imparato:

- il concetto di array, ovvero di un insieme di variabili destinate allo stesso scopo, cui è attribuito un nome unico, ma alle quali si può fare riferimento singolarmente tramite un indice
- la realizzazione BASIC di array tramite l'istruzione DIM
- l'accesso ad elementi di array mediante indici
- i concetti di INTERPRETE e COMPILATORE

GLI ACCORDI

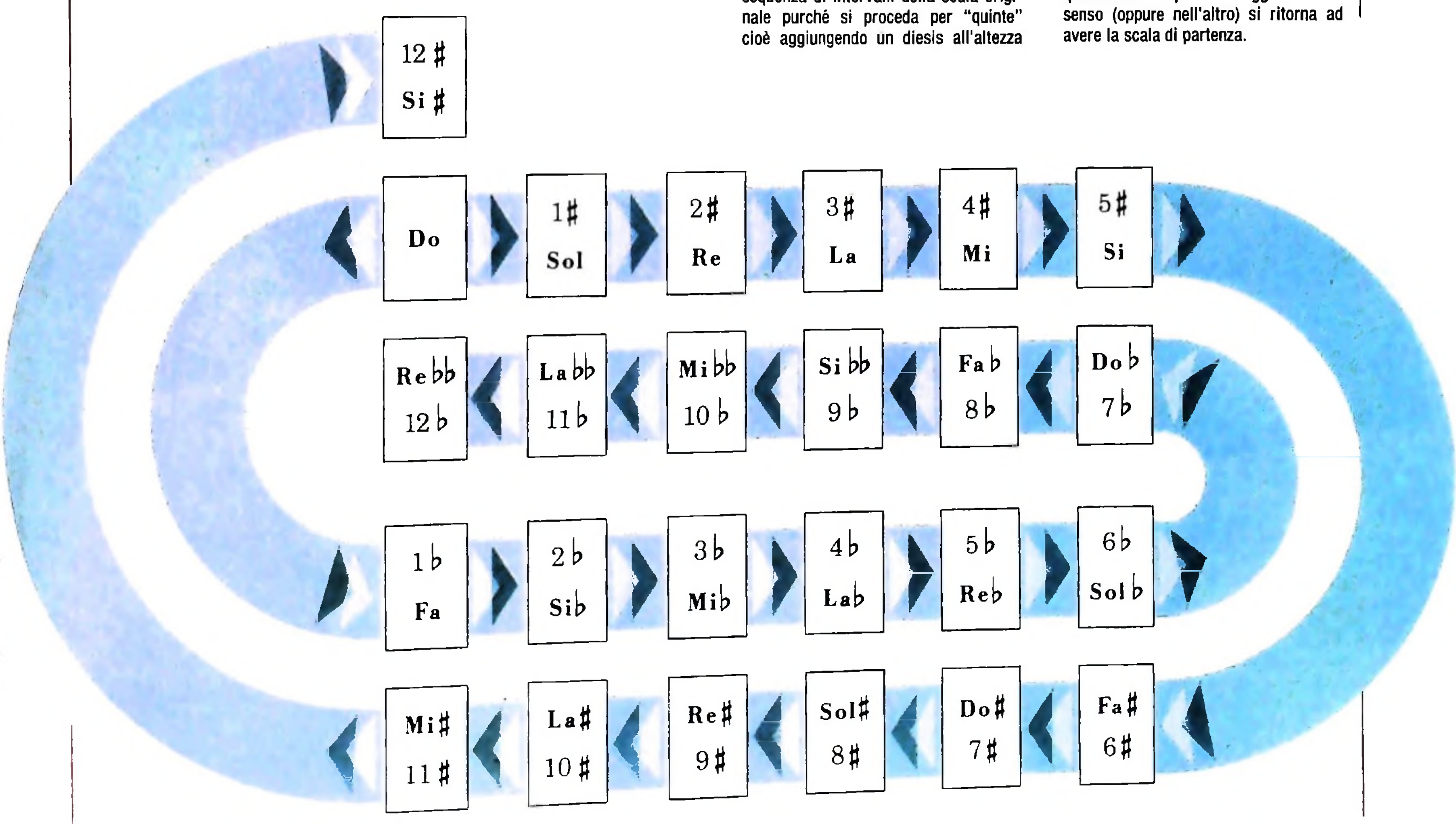
Sono la sovrapposizione di due o più suoni e determinano l'organizzazione "verticale" dei brani musicali.

Abbiamo visto come articolare orizzontalmente un testo musicale; ora vedremo invece come organizzarlo verticalmente. Parleremo quindi della sovrapposizione di suoni (due o più), cioè degli *accordi*. Prima di discutere gli accordi, però, ricordiamo una nozione fondamentale, quella di *tonalità*.

La tonalità

La scala diatonica, che abbiamo descritto precedentemente, si riferisce implicitamente alla tonalità di *do maggiore*: questo significa che l'*altezza centrale* del brano è il *do* e che la sequenza di intervalli è quella che caratterizza la scala diatonica (coincidente con la sequenza di intervalli del *modo maggiore*, che verrà discusso prossimamente). Tonalità è quindi comporre i brani musicali usando solamente le altezze di un certo insieme di altezze legate tra loro dai rapporti intervallari di un modo. Pertanto, rimandando come si è detto la trattazione dei modi, possiamo considerare le possibili combinazioni di altezze facendo riferimento agli intervalli della scala temperata.
Sul pentagramma descriviamo la tonalità mediante l'indicazione dei *diesis* e dei *bemolle* corrispondenti alle alterazioni necessarie per mantenere la sequenza di intervalli della scala

Qui sotto possiamo vedere il "circolo delle quinte": aggiungendo diesis (o bemolli) alle altezze della scala diatonica si ottengono scale che mantengono la sequenza di intervalli della scala originale purché si proceda per "quinte" cioè aggiungendo un diesis all'altezza corrispondente ad un intervallo di quinta ascendente (oppure aggiungendo un bemolle all'altezza corrispondente ad un intervallo di quinta discendente). In questo modo dopo dodici aggiunte in un senso (oppure nell'altro) si ritorna ad avere la scala di partenza.



diatonica, una volta fissata l'*altezza della tonica* (cioè l'altezza che determina la tonalità). Ad esempio, la tonalità di *do maggiore* prevede le altezze *do-re-mi-fa-sol-la-si*, mentre la tonalità di *re maggiore* prevede le altezze *re-mi-fa diesis-sol-la-si-do diesis* (ricordiamo che la sequenza di intervalli da mantenere è 2-2-1-2-2-2-1, espressa in semitoni). Nella figura della pagina precedente possiamo vedere schematicamente come vengono introdotti i diesis e i bemolli nelle tonalità: è il *circolo delle quinte*, così chiamato poiché ogni aggiunta di una alterazione (diesis o bemolle) corrisponde all'innalzamento o all'abbassamento di una *quinta* della tonica.

Triadi

La combinazione di tre suoni simultanei appartenenti a una certa tonalità, ottenuti come sequenza di due intervalli di terza, determina una *triade*; è un accordo costituito da un'altezza base (più grave), chiamata *fondamentale*, e dalla terza (ascendente) e dalla quinta (ascendente) relativa. Per chiarire sinteticamente quali sono i differenti casi di triadi che possiamo formare, consideriamo la seguente classificazione:
a) *triade maggiore*: sequenza di un intervallo di terza maggiore e di un intervallo di terza minore (4-3, in semitoni);
b) *triade minore*: sequenza di un intervallo di terza minore e di un intervallo di terza maggiore (3-4);
c) *triade diminuita*: sequenza di due intervalli di terza minore (3-3);
d) *triade eccedente*: sequenza di due intervalli di terza maggiore (4-4).
Notiamo che nei casi a) e b) la terza altezza dell'accordo è una quinta rispetto alla fondamentale, mentre nel caso c) è una quinta diminuita di un semitono (6 semitoni invece di 7) e nel caso d) è una quinta eccedente (8 semitoni invece di 7).
Inoltre, a seconda della posizione assoluta delle altezze di una triade, abbiamo i *rivolti*, determinati dalla posizione del suono più grave della triade; distinguiamo quindi i tre possibili rivolti, come si può vedere nell'illustrazione:
a) *accordo fondamentale*, quando il suono più grave è la fondamentale;
b) *I rivolto*, quando il suono più grave è la terza; in questo caso la quinta viene a trovarsi (superiormente) a un intervallo di terza e la fondamentale a un intervallo di sesta;.perciò il I rivolto è anche chiamato *accordo di terza e sesta*;
c) *II rivolto*, quando il suono più grave è la quinta; in questo caso la fondamentale è una quarta sopra e la terza è una sesta sopra; il II rivolto è perciò detto *accordo di quarta e sesta*.
Possiamo costruire triadi su tutti i gradi di una scala (cioè a partire da una qualunque altezza di una scala) semplicemente componendo l'accordo come sequenza di due terze consecutive della scala; se indichiamo con I il primo grado, con II il secondo grado ecc. abbiamo le seguenti triadi:
— triade maggiore sul I grado (I-III-V): *tonica*;
— triade minore sul II grado (II-IV-VI): *sopratonica*;
— triade minore sul III grado (III-V-VII): *mediante*;
— triade maggiore sul IV grado (IV-VI-I): *sottodominante*;
— triade maggiore sul V grado (V-VII-II): *dominante*;
— triade minore sul VI grado (VI-I-III): *sopradominante*;
— triade diminuita sul VII grado (VII-II-IV): *sensibile* (o *settima*).
Per ognuno di questi sono poi possibili i rivolti; abbiamo quindi sette triadi con quattordici rivolti costruiti sui gradi di

Qui sopra vediamo rappresentate le posizioni dei rivolti di un accordo (con fondamentale il do). In alto, le triadi nella tonalità del "do maggiore".

una particolare scala diatonica, ovvero di una certa tonalità maggiore.
Nell'armonia tradizionale troviamo criteri che guidano alla realizzazione di sequenze di accordi costruiti sui gradi di una scala (come già abbiamo visto per le sequenze di intervalli nelle melodie); una triade può precedere o seguire un'altra triade in dipendenza dai rapporti intervallari delle altezze degli accordi stessi; in particolare, se due o tre altezze di un accordo sono più acute delle altezze dell'accordo precedente abbiamo un *moto retto*, se un'altezza di un accordo è più acuta ed un'altra è più grave delle corrispondenti dell'accordo precedente abbiamo un *moto contrario*, se infine un'altezza sale o scende e un'altra resta ferma abbiamo un *moto obliquo*. Dobbiamo tener conto anche di altri divieti dettati dal co-

Suoniamo Mozart con M10

Abbiamo già visto che possiamo realizzare l'esecuzione di una scala in una certa varietà di modi differenti usando un elaboratore. Proviamo ora a trascrivere una semplice melodia usando (per ora) l'istruzione SOUND ed eventualmente le strutture di controllo del linguaggio BASIC; il testo con cui facciamo questa prova è l'inizio dell'ouverture delle *Nozze di Figaro* di W.A. Mozart, che riportiamo qui sotto. Prossimamente vedremo insieme alcuni modi per descriverlo ed eseguirlo, ma nel frattempo potete provare da voi.

Il libretto dell'opera *Le nozze di Figaro*, di Mozart, rappresentata per la prima volta a Vienna nel 1786 mune buon gusto musicale: per esempio quello di usare intervalli di quinta e di ottava parallelamente. Sulla base della prassi musicale tonale possiamo quindi tentare una tabella dei collegamenti preferenziali tra le triadi, come possiamo vedere qui a lato.

Regole di preferenza nella successione delle triadi

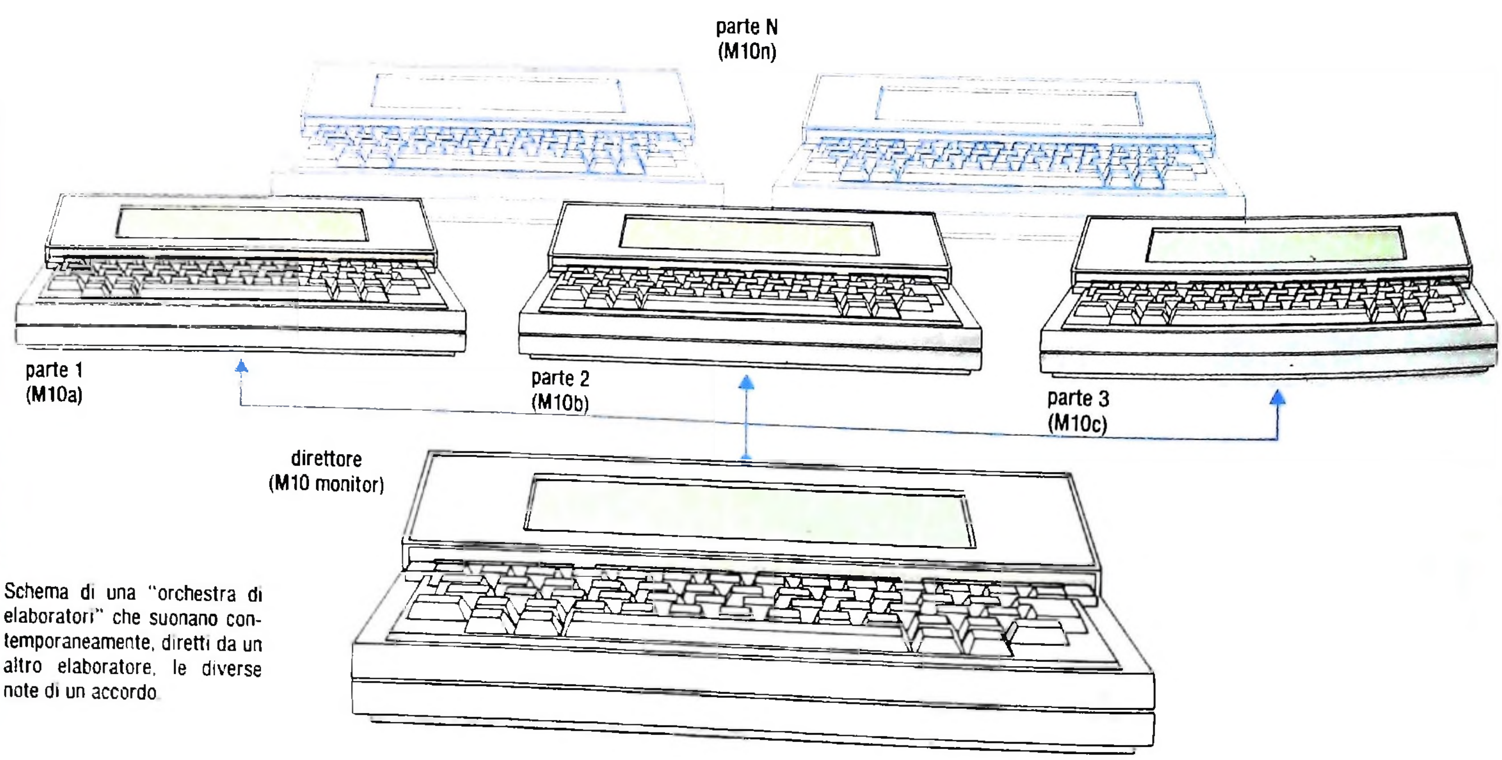
Accordo	collegamenti con altri accordi (con preferenza secondo l'ordine)
I (tonica	qualsiasi accordo
II (sopratonica)	V III IV VI VII
III (mediante)	VI IV II V
IV (sottodominante)	V I VI II VII III
V (dominante)	I VI III IV
VI (sopradominante)	II V IV III
VII (sensibile)	I VI III V

Accordi di quattro o più note

Per costruire accordi più complessi aggiungiamo ulteriori intervalli di terza (minore o maggiore). Dalla triade con la fondamentale, la sua terza e la sua quinta costruiamo un *accordo di settima* aggiungendo la settima, come il nome suggerisce, l'*accordo di nona* aggiungendo la seconda nell'ottava superiore (appunto, la nona), l'*accordo di undicesima* aggiungendo la quarta dell'ottava superiore (l'undicesima) ecc. Quanto più procediamo nella complicazione dell'accordo e tanto più l'accordo assume caratteristiche di dissonanza; accordi di undicesima o tredicesima sono molto usati nella musica "classica", ma ancor più nella musica jazz.

Realizzazione degli accordi su elaboratore

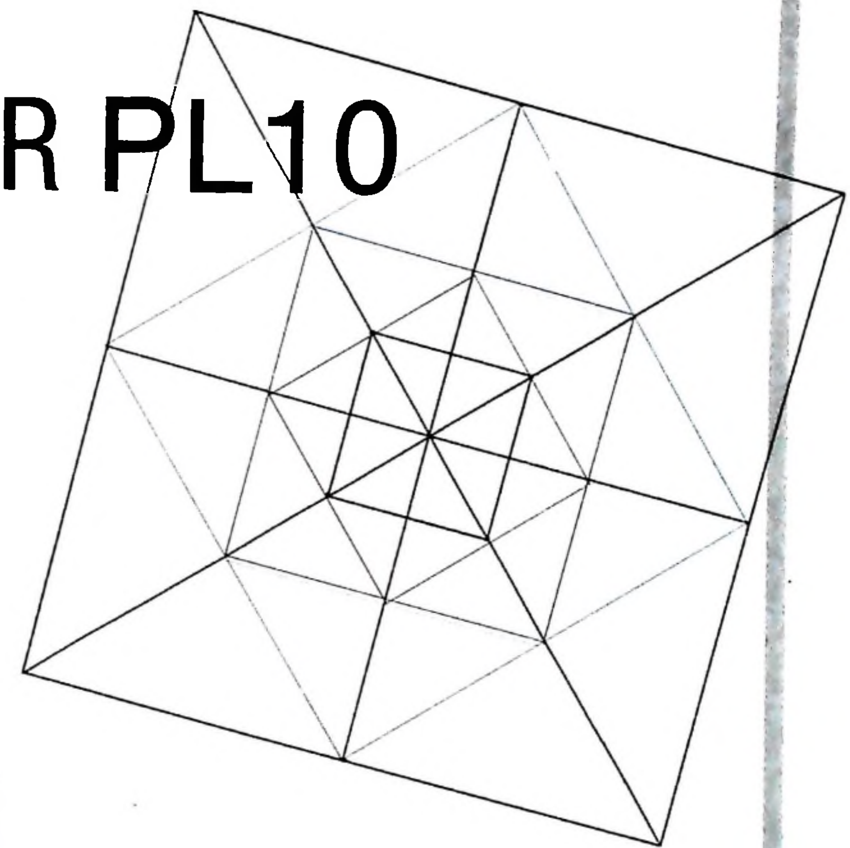
Per realizzare accordi con un elaboratore possiamo seguire due metodi: o guardiamo l'accordo come una singola entità, cioè la partitura come una sola sequenza di accordi complessi quanto si vuole, o consideriamo una successione di accordi come la sovrapposizione nel tempo di più processi musicali e quindi l'accordo non come un elemento costitutivo, ma come sovrapposizione di più parti musicali.
Questa seconda impostazione è molto vicina alla prassi oltre che alla logica della composizione: possiamo infatti immaginare che ogni parte (cioè ogni linea melodica) corrisponda a uno strumento musicale monofonico (cioè che produce un solo suono per volta).
In questo modo l'accordo corrisponde al verificarsi di una certa situazione armonica dipendente dall'andamento delle singole parti.
Per realizzare l'accordo occorre uno strumento *polifonico* (in grado di produrre più di un suono contemporaneamente) oppure più strumenti monofonici coordinati come gli strumentisti di un'orchestra, per esempio, un insieme di M10, a ognuno dei quali affidiamo una parte. Il problema che ci rimane è il coordinamento, ovvero la *sincronizzazione* tra le diverse parti.
Ecco allora, per restare nella metafora, la necessità di un *direttore* dell'orchestra di elaboratori che, naturalmente, può a sua volta essere un elaboratore.

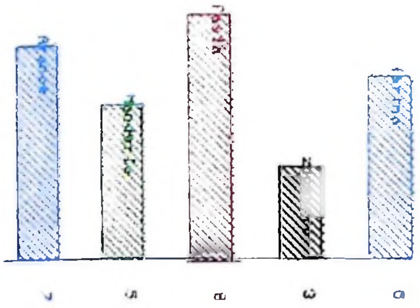


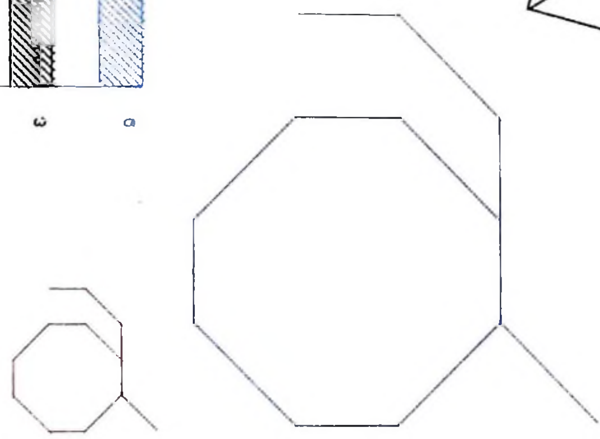
Schema di una "orchestra di elaboratori" che suonano contemporaneamente, diretti da un altro elaboratore, le diverse note di un accordo.

IL MICROPLOTTER PL10

Una unità di uscita piuttosto versatile con numerose applicazioni per la grafica.







Con il microplotter PL 10 è possibile realizzare testi come con una stampante, e inoltre grafici, disegni e schemi di vario genere fra cui istogrammi.Qui sopra possiamo vedere quadrati di diversi colori contenuti gli uni negli altri, realizzabili mediante il programma relativo rappresentato più avanti.

Abbiamo precedentemente detto come il microplotter PL10 possa essere utilizzato sia come stampante sia come plotter.
Le due modalità d'impiego richiedono comandi diversi, ma un'unica istruzione BASIC è utilizzata per inviarli al PL10: l'istruzione LPRINT.
Le due modalità si chiamano: Text Mode (che viene automaticamente selezionata all'accensione dello strumento) e Graphic Mode.
Si può passare dalla modalità Text a quella Graphic dando l'istruzione: LPRINT CHR$(18).
Viceversa, per ritornare al Text Mode, si dà l'istruzione: LPRINT CHR$(17).
Il Text Mode è la modalità necessaria per utilizzare il PL10 come stampante.
Il Graphic Mode serve invece per fare disegni con il plotter utilizzando alcuni comandi speciali che specifichiamo qui di seguito. Tutti i comandi andranno considerati come stringhe di caratteri e quindi posti fra virgolette nell'istruzione BASIC. Esempio: LPRINT "A", serve per riportare il microplotter nella modalità Text, posizionando la penna sul margine sinistro del foglio.
Volendo invece, sempre per esempio, portare la penna nell'origine senza scrivere, bisogna usare l'istruzione LPRINT "H", con il comando al plotter ancora tra virgolette.

Comandi per la modalità grafica

In questi comandi si considerano le coordinate x e y riferite agli assi orizzontale e verticale. La distanza fra i bordi della carta è di 480 punti indirizzabili. Il PL10 accetta lo stesso intervallo di valori per x e y, ma quelli superiori a 480 per x sono trattati come se fossero uguali a 480.

CHR$(17)
- Riporta nella modalità Text.

A
- Riporta nella modalità Text e posiziona la penna sul margine sinistro.

Cn
- Cambia colore: il colore è determinato dal valore di n, che può essere 0 = nero, 1 = rosso, 2 = verde, 3 = blu.

Questa codifica vale soltanto se le penne sono state installate nella cartuccera con le posizioni indicate.

Dx1,y1,x2,y2,x3,y3...
- Disegna: questa istruzione disegna una linea dalla posizione attuale alle coordinate specificate da x1,y1, rispetto all'o-

rigine, che è posta sul bordo del lato sinistro della carta. Poi disegna una linea da x1,y1 a x2,y2 ecc. I valori di *x* e *y* possono essere compresi fra 0 e 999.

Vediamo subito un piccolo esempio per questi comandi iniziali:

```
10 REM Entra in modalità grafica
20 LPRINT CHR$(18)
30 REM seleziona il colore verde
40 LPRINT "C2"
50 REM disegna un quadrato
60 LPRINT "D0,25,25,25,25,0,0,0
70 REM Ritorna in modalità Text
80 LPRINT "A"
90 END
```

H
- **Home** (casa): serve per posizionare la penna nell'origine senza tracciare una linea.

Mx,y
- **Move** (muovi la penna su): sposta la penna senza tracciare una linea nella posizione specificata dalle coordinate *x*, *y*, rispetto all'origine.

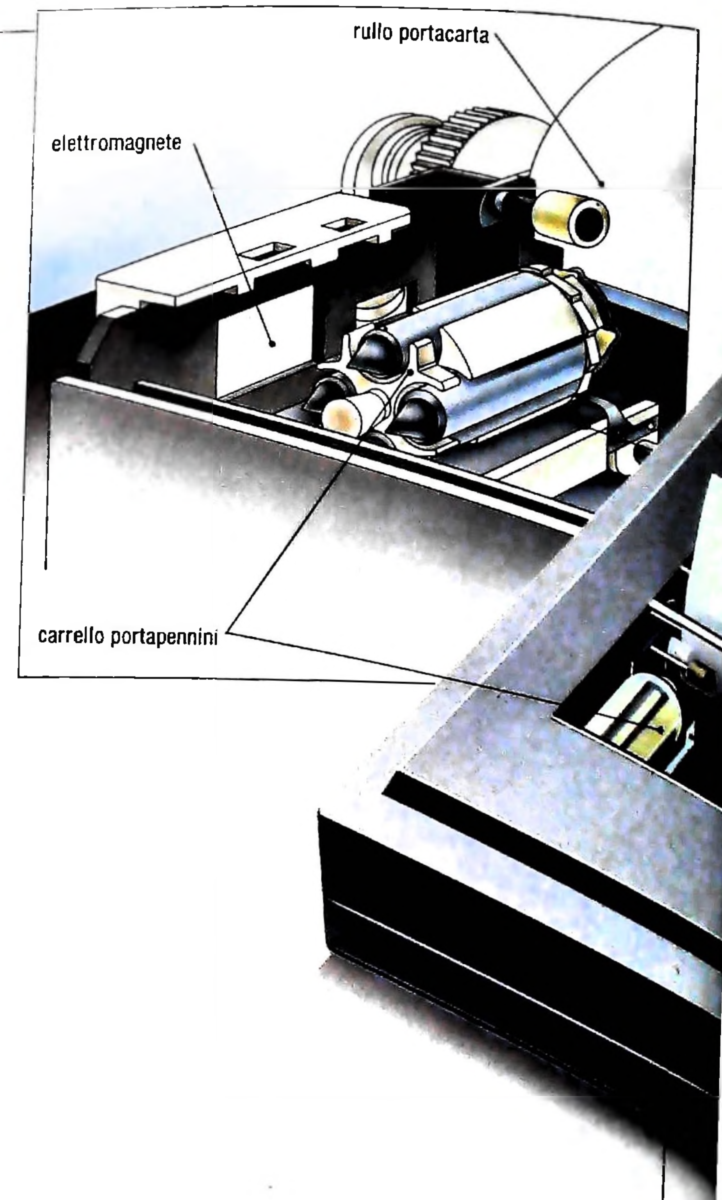
I
- **Inizializza**: serve per ridefinire l'origine del sistema di riferimento nel punto in cui si trova la penna.

Esempio:

```
10 LPRINT CHR$(18)
20 REM Muove la penna senza disegnare
25 REM nel punto 100,100
30 LPRINT "M100,100"
40 REM Disegna un segmento
50 LPRINT "D100,100,300,300"
60 REM Pone l'origine nel punto 300,300
70 LPRINT "I"
80 REM Muove la penna senza disegnare
90 REM nel punto 100,100 che ora e'
100 REM diverso dal precedente.
110 LPRINT "M100,100"
120 REM Muove la penna nell'origine.
130 LPRINT "H"
140 END
```

Jx1,y1,x2,y2,x3,y3,...
- **Disegna in modo relativo**: agisce come D, ma le coordinate sono riferite alla posizione attuale della penna. I valori di *x* e *y* possono essere fra -999 e 999.

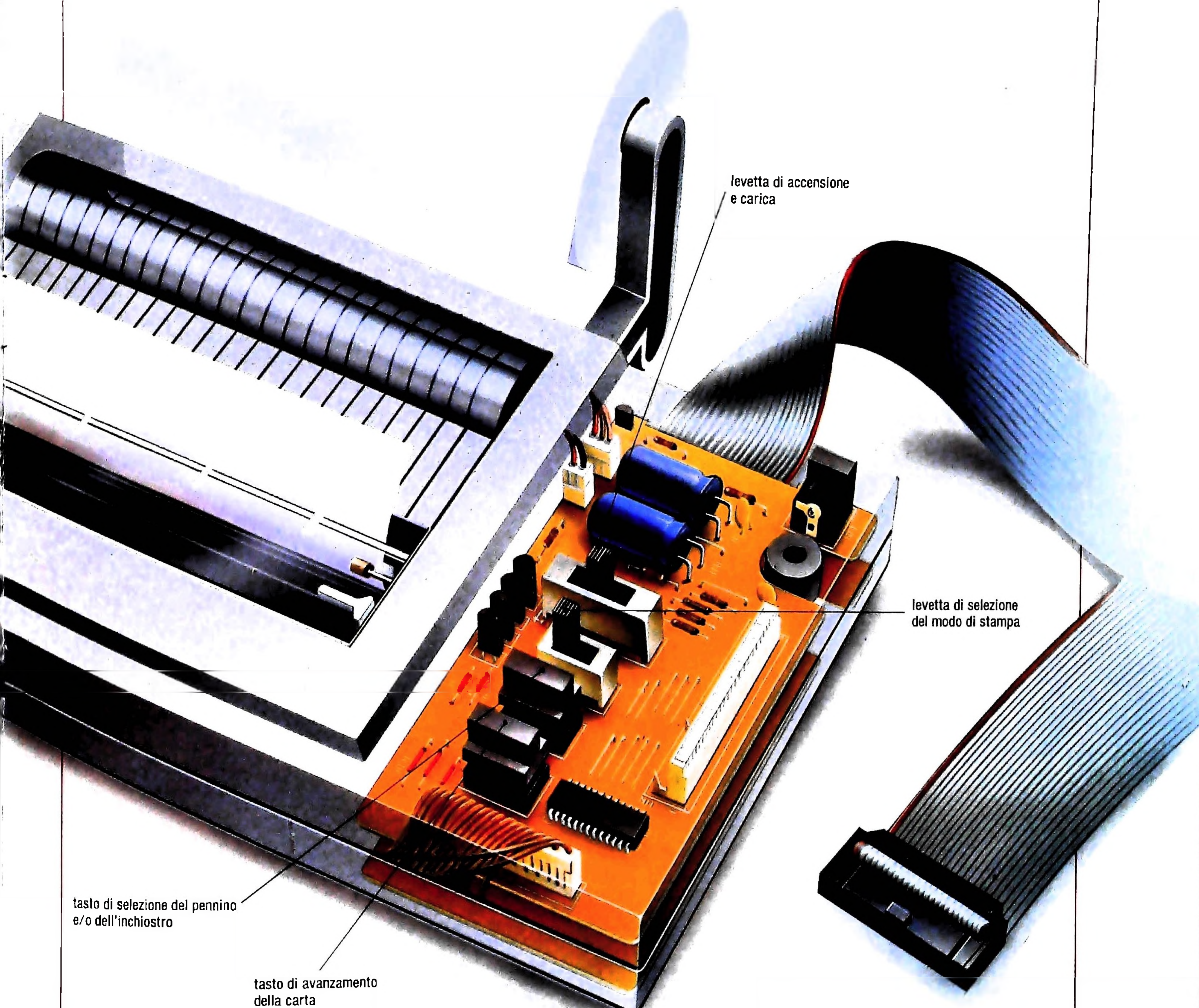
Per esempio:
LPRINT "J0,30,30,0,0,-30,-30,0" disegna un quadrato.



Il microplotter PL 10 della Olivetti, con il suo cavo "pin to pin" per collegarlo al calcolatore M 10. In alto vediamo un particolare dell'interno, rivelatore della complessità logica e meccanica di una periferica che pure risulta apparentemente semplice per l'utente.

Rx,y
- **Move relative** (muovi la penna su *x*,*y* in coordinate relative): muove la penna senza tracciare la linea dalla posizione attuale della penna alle coordinate *x*,*y* riferite al punto in cui si trova la penna stessa. I valori di *x* e *y* possono essere compresi nell'intervallo da -999 a 999.

Xasse, incremento, volte
- **Disegna assi**: serve per disegnare gli assi *x* e *y*, specificati dai codici 1 per x e 0 per y. L'incremento fra le tacche sul-



l'asse può essere compreso fra -999 e 999. Il numero di volte che "incremento" deve essere ripetuto è dato da "volte", che deve essere compreso fra 1 e 125.

Esempio:

```
10 LPRINT CHR$(18)
20 LPRINT "C1"
30 REM disegna l'asse delle X
40 LPRINT "X1,50,8"
50 LPRINT "H"
60 REM disegna l'asse delle X
70 LPRINT "X0,50,8"
80 LPRINT "H"
90 REM Cambia colore
100 LPRINT "C2"
110 REM disegna un grafico
120 LPRINT
"D0,0,50,50,100,150,150,300,200,250,250,
```

```
50,300,100,350,400,400.250"
130 REM Ritorna nell'origine utilizzan-
140 REM do l'istruzione R
150 LPRINT "R-400,-250"
160 REM Cambia colore
170 LPRINT "C3"
180 REM Disegna un altro grafico usando
190 REM il comando J
200 LPRINT "J0,0,50,70,50,130,50,-30,50,
-50,50,100,50,100,50,-36,50,67"
210 LPRINT "H"
220 END
```

P stringa
• Print (stampa): questo comando stampa i caratteri alfanumerici specificati in "stringa".

S dimensione
• Cambia la dimensione dei caratteri stampabili da P. Il valore della dimensione può variare da 0 (80 caratteri per linea) a 63 (1 carattere per linea).
Se non viene data altra specifica, vale automaticamente 0.

Q n
• Stampa in una direzione: questo comando specifica la direzione di stampa a seconda del valore di n, che può essere 0 (da sinistra a destra), 1 (dal basso all'alto sulla destra della carta), 2 (da sinistra a destra capovolta), 3 (dall'alto al basso sulla sinistra della carta). All'accensione dello strumento il valore di n è 0.

Esempio:

```
10 LPRINT CHR$(18)
20 LPRINT "H"
30 LPRINT "C2"
40 LPRINT "Q 3"
50 LPRINT "S 20"
60 LPRINT "Pciao"
70 LPRINT "H"
80 END
```

Va segnalato che per n = 1 oppure 2 bisogna che la penna sia posizionata sulla destra del carrello: ovvero bisogna introdurre nel programma precedente l'istruzione:

```
35 LPRINT "M480,0"
```

Quadrati di diversi colori

Precedentemente abbiamo visto rappresentata una serie di quadrati di differenti colori contenuti uno negli altri che sono stati disegnati dal microplotter. Possiamo provare un possibile programma, come vediamo qui sotto, che realizza il disegno di cui si è accennato.

```
10 LPRINT CHR$(18)
20 LPRINT "H"
40 LPRINT "D0,0,0,400,400,400,400,0,0,0"
50 LPRINT "C1"
60 LPRINT "M200,0"
70 LPRINT
"D200,0,0,200,200,400,400,200,200,0"
80 LPRINT "C2"
90 LPRINT "M100,100"
100 LPRINT
"D100,100,100,300,300,300,300,100,100,10
0"
110 LPRINT "C3"
120 LPRINT "M200,100"
130 LPRINT
"D200,100,100,200,200,300,300,200,200,10
0"
135 LPRINT "C0"
140 LPRINT "M150,150"
145 LPRINT
"D150,150,150,250,250,250,250,150,150,15
0"
150 LPRINT "M0,400"
160 LPRINT "D0,400,400,0"
170 LPRINT "I"
180 LPRINT "M0,400"
190 LPRINT "D0,400,-400,0"
200 LPRINT "M-400,200"
210 LPRINT "D-400,200,0,200"
220 LPRINT "M-200,400"
230 LPRINT "D-200,400,-200,0"
290 LPRINT "H"
300 END
```

PERSONAL COMPUTER OLIVETTI M20

PERSONAL COMPUTER OLIVETTI. UNA FAMIGLIA CHE CRESCE

In soli due anni più di 25.000 persone in Italia hanno scelto M20, il personal computer con oltre 700 programmi pronti.
Sono dati che confermano la leadership Olivetti anche nel settore dei personal. Sono dati che continuano a crescere. Come cresce, in macchine e programmi, la famiglia dei personal computer Olivetti: una linea di modelli che, per rispondere a esigenze diverse, offre, oltre a differenti capacità di memoria, un'ampia scelta di sistemi operativi (MS-DOS, CP/M-86, PCOS, UCSD-P). Sono personal computer con tecnologia a 16 bit e capacità di "communication", progettati quindi per essere validi anche domani e per integrarsi agevolmente nelle strutture di elaborazione dati e di automazione dell'ufficio presenti e future. Perché Olivetti protegge i vostri investimenti in macchine e programmi.

Con i personal computer Olivetti M20 il lavoro individuale diventa più semplice e produttivo. Constatarlo è semplice. Basta rivolgersi a uno degli oltre 500 punti di vendita e assistenza M20. Lì il vostro prossimo personal vi aspetta per una prova.

Anche in leasing con Olivetti Leasing S.p.A.

MS-DOS, marchio registrato Microsoft Inc.
CP/M-86, marchio registrato Digital Research
UCSD-P System, marchio registrato Regent of the University of California

olivetti

l'universo della comunicazione