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

14 CORSO PRATICO COL COMPUTER

421669

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

diretto da GIANNI DEGLI ANTONI

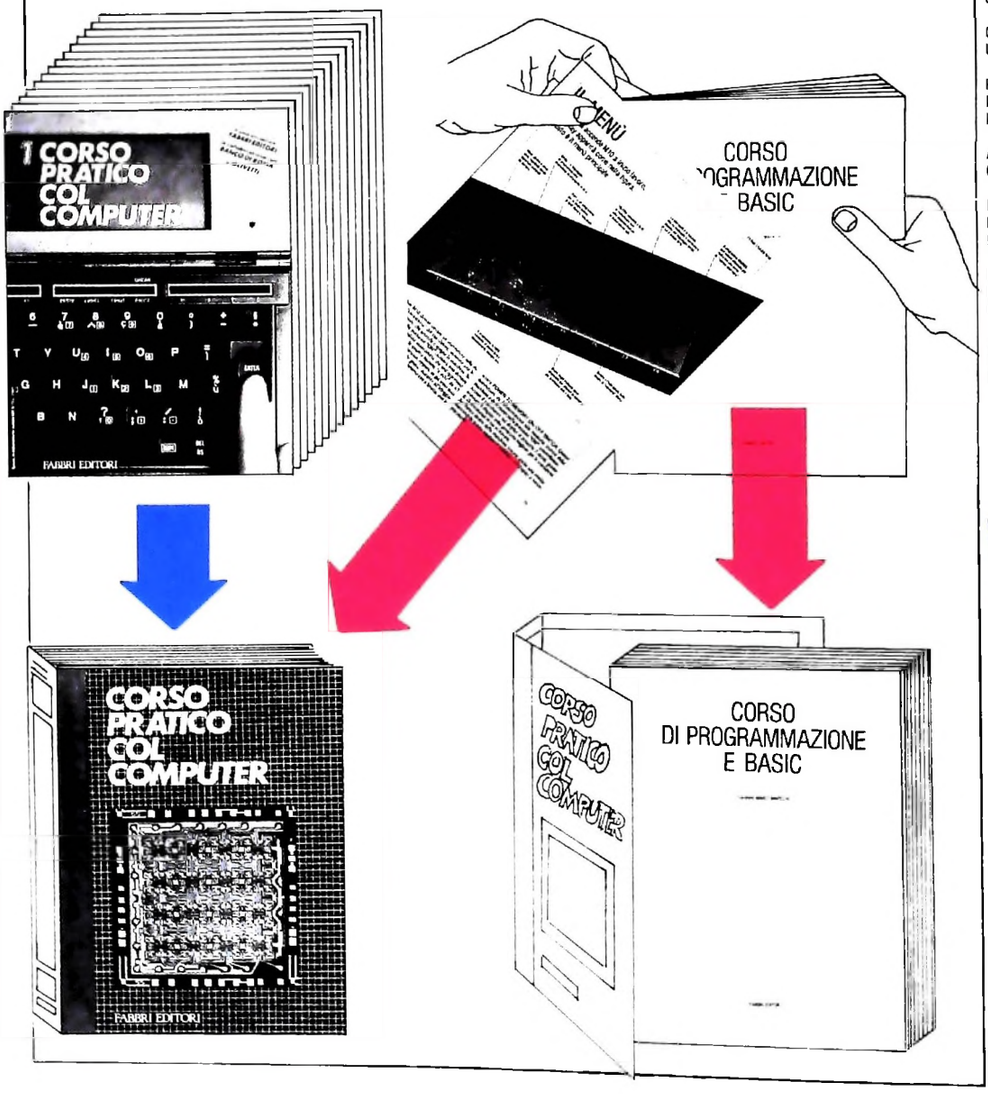
COME TRASPORRE UN BRANO MUSICALE MEDIANTE UN PROGRAMMA

FABBRI EDITORI

AVVISO AI LETTORI

Con questo fascicolo si conclude il primo volume del "Corso pratico col computer". Per rilegare il volume si useranno:

- i fascicoli dal n. 1 al n. 14;
- occorrerà staccare l'inserto centrale relativo al "Corso di programmazione e BASIC" a partire dal fascicolo n. 2; il primo inserto è costituito di 8 pagine e gli inserti successivi di 4 pagine ciascuno; tutti gli inserti contenuti nei fascicoli, fino al n. 72, dovranno essere conservati per essere rilegati nel volume "Corso di Programmazione e BASIC", la cui copertina sarà messa in vendita con il fascicolo n. 30;
- i risguardi, inseriti nella copertina del volume;
- la copertina del volume, che è stata messa in vendita con il n. 10
- ricordiamo che il frontespizio del volume fa parte del fascicolo n. 1 e il sommario fa parte del fascicolo n. 14.



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
Eidos (TIZIANO BRUGNETTI, SANDRO MISSAGLIA), ADRIANO DE LUCA, GOFFREDO HAUS, ENNIO PROVERA, Etnoteam (ADRIANA BICEGO)

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

Direttore Editoriale
ORSOLA FENGHI

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 BREGGÈ

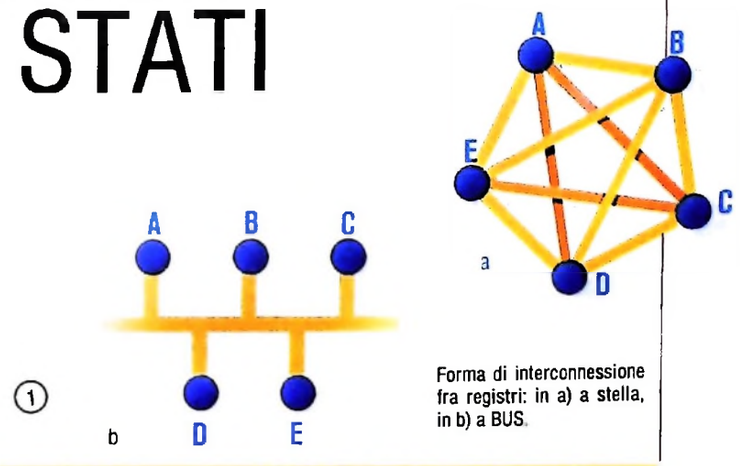
Segretarie di Redazione
RENATA FRIGOLI
LUCIA MONTANARI

NEL PROSSIMO NUMERO

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

I CIRCUITI A TRE STATI O THREE-STATE

Come ampliare l'algebra di Boole.



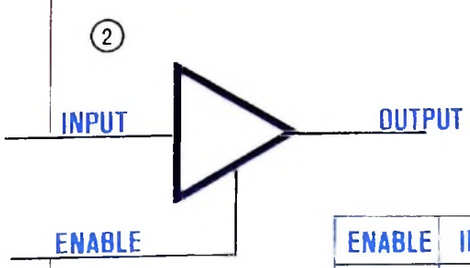
Forma di interconnessione fra registri: in a) a stella, in b) a BUS.

* Con questo termine viene denominato qualsiasi circuito elettronico digitale avente determinate caratteristiche. Esempi di registri sono i circuiti di memoria, controllo, ALU, accumulatori ecc.

Pensiamo ora di voler interconnettere fra di loro cinque registri* (si veda la nota nel riquadro a lato).
Ciò è possibile in due modi diversi, mostrati in figura 1a) e 1b), di cui ora esaminiamo le caratteristiche. Nella forma a) esistono 10 canali di trasmissione, quindi sono necessarie molte "piste". La velocità di trasmissione dei dati è molto alta, perché c'è la possibilità che due registri, nel nostro caso, possano trasmettere contemporaneamente sempre e quando i registri ricevitori non sono gli stessi per ambedue. Naturalmente una struttura del genere comporta la costruzione di un registro di controllo molto complesso.
Nella forma b), invece, abbiamo un solo canale per tutti i registri, tuttavia ad ogni dato istante esiste solo un registro che può trasmettere, mentre tutti gli altri possono ricevere, al limite.
Le due configurazioni, come tutte quelle intermedie, sono usate nelle architetture dei microprocessori in misura variabile: tutto dipende dalle caratteristiche che si desiderano. A ogni modo, la configurazione attualmente di gran lunga più usata nei microprocessori è quella della figura 1b, sia per il costo minore sia per la minor complessità del sistema di controllo.
Come si era visto nei circuiti precedenti, in cui più o meno si cercava di realizzare la configurazione b), esisteva il problema delle interferenze possibili sulla linea del BUS per i diversi registri trasmettitori. Si era cercato di ovviare a ciò inserendo dei circuiti a collettore aperto, però bisognava sempre tener sotto controllo tutti i registri collegati al BUS perché mantenessero sempre la loro uscita in circuito aperto.
Questo metodo crea non pochi problemi, se si pensa a tutto quello che possono fare gli altri registri mentre uno di essi si trova in fase di trasmissione o di ricezione.

Il terzo stato

La soluzione di questo problema è stata trovata nella creazione di circuiti a tre stati (detti "three-state"), cioè dei circuiti che, oltre ad avere i due stati logici booleani 0 (falso) e 1 (vero) hanno anche un terzo stato di "alta impedenza", che non carica in alcun modo il BUS, mentre il registro collegato ad esso è isolato elettricamente. Il simbolo del circuito three-state è mostrato nella figura 2 assieme alla sua tabella di verità e allo schema della sua struttura interna. L'uso di questi

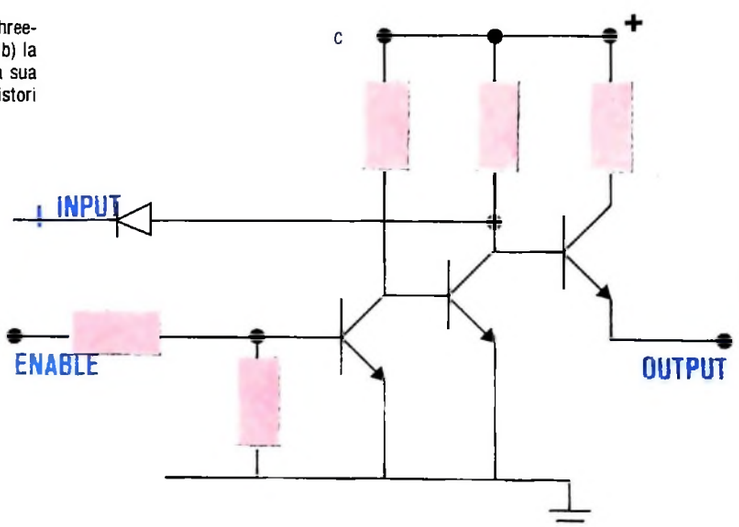


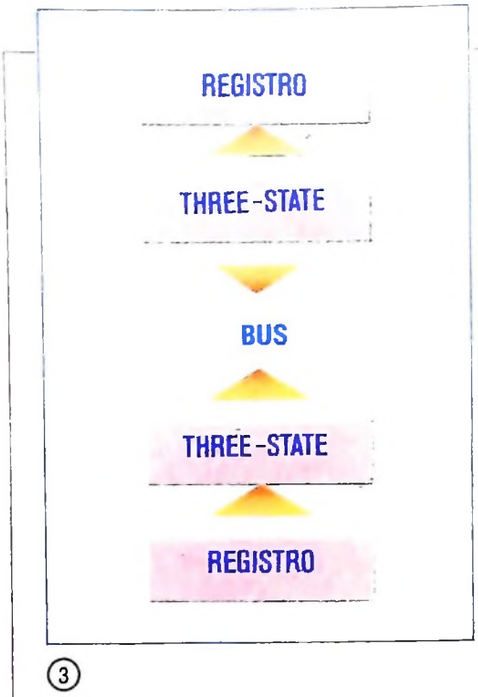
Il circuito a tre stati (three-state). In a) il simbolo, in b) la tabella di verità, in c) una sua realizzazione con transistori (le tre sbarre verticali).

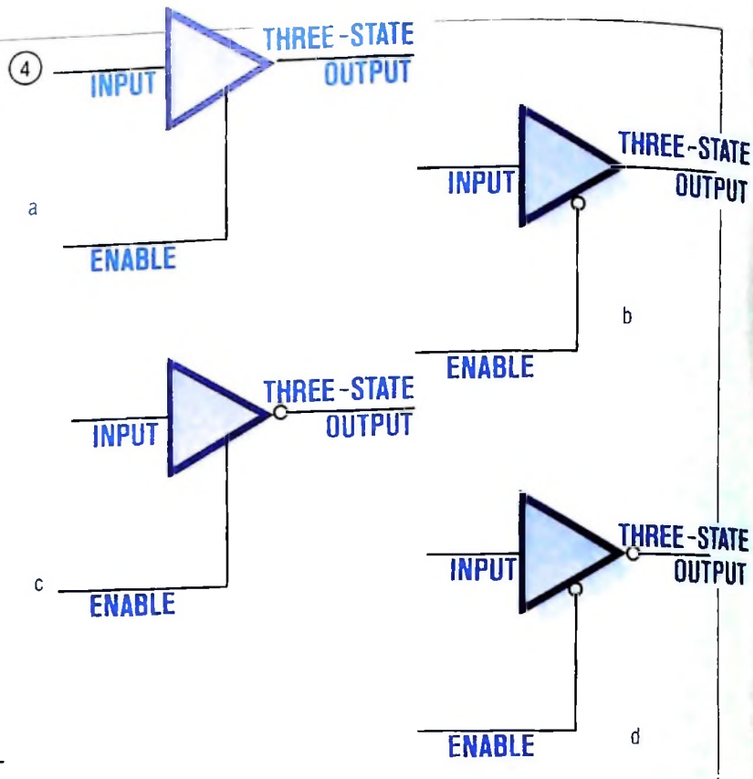
ENABLE	INPUT	OUTPUT
0	0	ALTA IMP (•)
0	1	ALTA IMP
1	0	0
1	1	1

b

(•) impedenza







Qui sopra, il circuito three-state come interfaccia fra un registro e un BUS. A destra, rappresentazione grafica dei circuiti a tre stati. In a), ATTIVO POSITIVO NON INVERSORE; in b), ATTIVO NEGATIVO NON INVERSORE; in c), ATTIVO POSITIVO INVERSORE; in d), ATTIVO NEGATIVO INVERSORE.

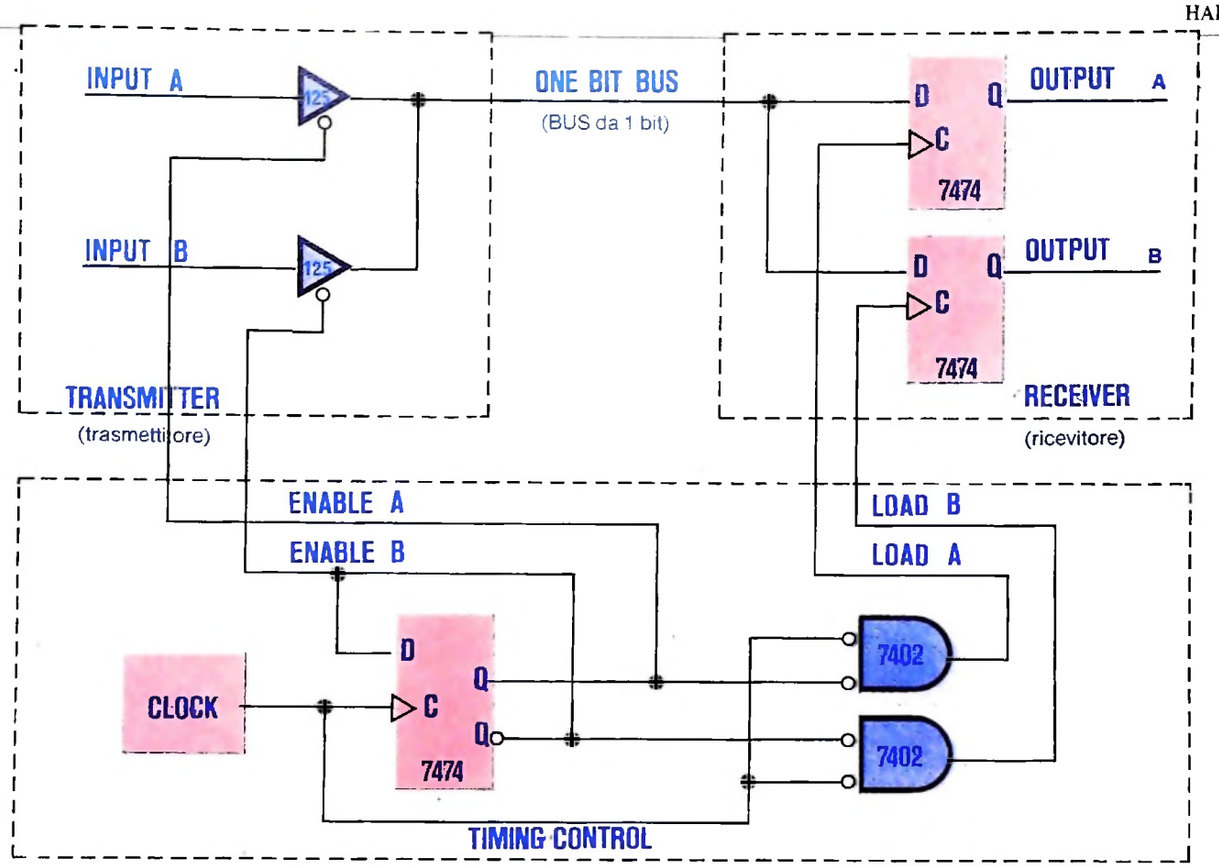
circuiti è a sé stante, per cui vengono messi come interfaccia fra i registri e il BUS, come mostra la figura 3.
Le realizzazioni in commercio sono indicate in figura 4: ciò che varia tra una configurazione e l'altra è, da un lato, il trattamento del segnale che lo inverte o lo lascia normale all'uscita, dall'altro il meccanismo di controllo che può abilitare o no il circuito con uno 0 o un 1, secondo la configurazione.
La figura 5 mostra un circuito analogo a quello illustrato nella figura 4 a pag. 199. Questa volta, però, al posto dei circuiti a collettore aperto abbiamo dei circuiti a tre stati. Ribadiamo ancora l'importanza dei due segnali ENABLE e LOAD. Con il primo permettiamo il passaggio dei dati al BUS in forma asincrona, cioè una volta ATTIVO, l'azione viene eseguita immediatamente. Con il LOAD la modalità è sincrona: i dati diventano ATTIVI con il segnale di clock.
Finora abbiamo visto modi diversi per costruire il circuito di controllo. Esso è quasi sempre un circuito sequenziale, composto da un circuito combinatorio, da una memoria e da un sistema di retroazione, che permette di realizzare lo stato presente in relazione con lo stato passato.
Analizziamo ora il circuito di figura 6; in esso si rispecchia uno schema classico di controllo di memoria RAM (lettura e scrittura) o ROM (solo lettura). La memoria complessiva RAM + ROM è uguale a 48 parole di 4 bit. Ogni blocco di memoria RAM ha due entrate di comando: CE e WE. La prima indica il comando che abilita la memoria a essere letta o scritta, mentre quando WE è 0 la memoria viene scritta e quando è 1 viene letta. Per la ROM c'è solo il comando di abilitazione, per cui, una volta ATTIVO, l'unica operazione possibile è la lettura.

Per poter accedere a 48 parole di memoria occorrono 6 linee di indirizzi. Con le linee 0, 1, 2 e 3 accediamo alle 16 parole interne ad ogni blocco. Con le 4 e 5 scegliamo, attraverso un decodificatore, quale dei tre blocchi viene abilitato. Facendo i conti vediamo che con 6 bit di indirizzo possiamo arrivare sino a 64 (cioè 2^6) locazioni di memoria per cui, nel nostro caso, ne avanzano 16 indefinite. Ne segue che una delle uscite del decodificatore, l'ultima a destra, è libera.**

** Spieghiamo brevemente cos'è un decodificatore. È un circuito elettronico come in figura 7, in cui ci sono due segnali in ingresso (in questo caso), per cui all'uscita abbiamo quattro diverse combinazioni possibili (2^2). Se le entrate fossero 4, avremmo 16 (cioè 2^4) possibili uscite diverse. Anche in questo caso, gli stati ATTIVI possono essere o POSITIVI o NEGATIVI secondo le necessità.

Analizziamo un po' i comandi cominciando con le linee di indirizzi. Queste devono essere abilitate prima di tutte le altre, per cui devono essere fisse e stabili per quando arriva il comando di lettura o scrittura. I comandi READ e WRITE, attivi negativi, sono esclusivi l'uno dell'altro: questi comandi devono essere gli ultimi a diventare attivi. I dati presenti nel BUS BIDIREZIONALE formato da quattro linee vengono sdoppiati alle entrate dei BUFFER in linee di scrittura e linee di lettura. I dati in scrittura non sono di tipo tre stati, poiché i dati che arrivano dal BUS non sono scritti in memo-

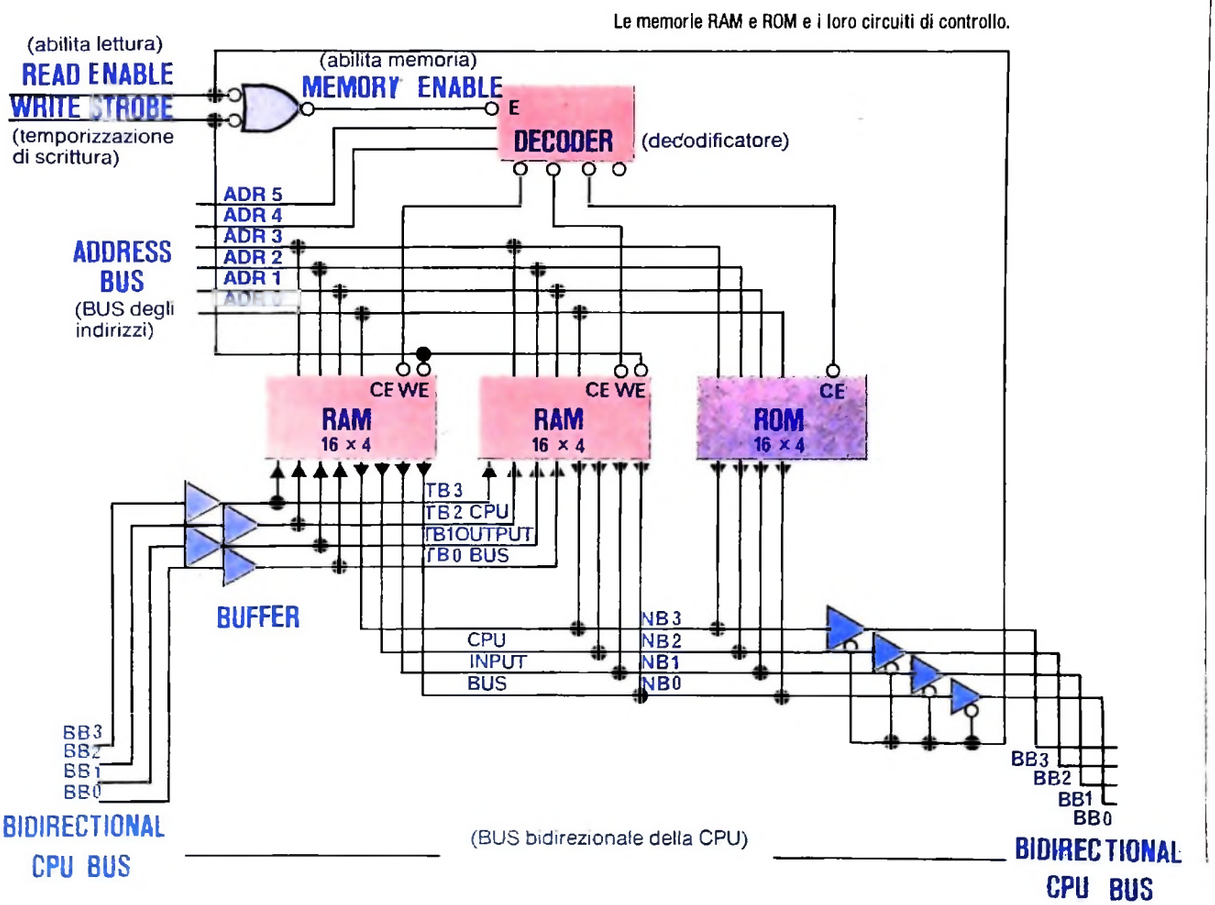
⑤

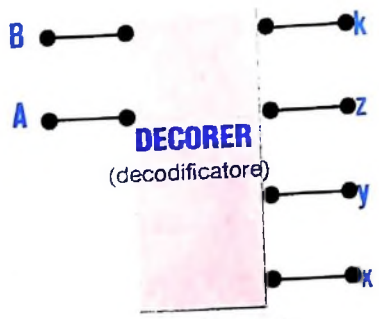


L'uso di un circuito a tre stati in un BUS da 1 bit.

⑥

Le memorie RAM e ROM e i loro circuiti di controllo.

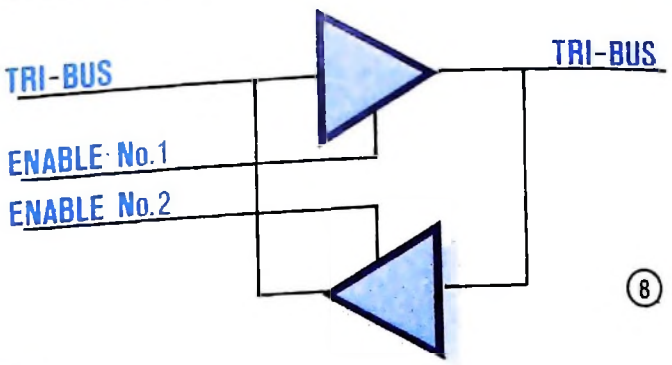




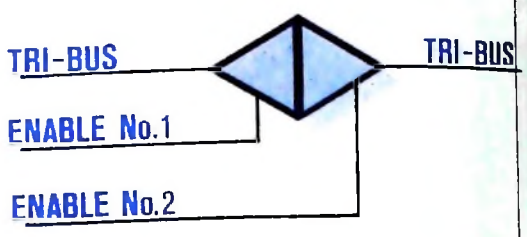
7

INPUT		OUTPUT			
A	B	x	y	z	k
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

Il decodificatore e la sua tabella di verità.

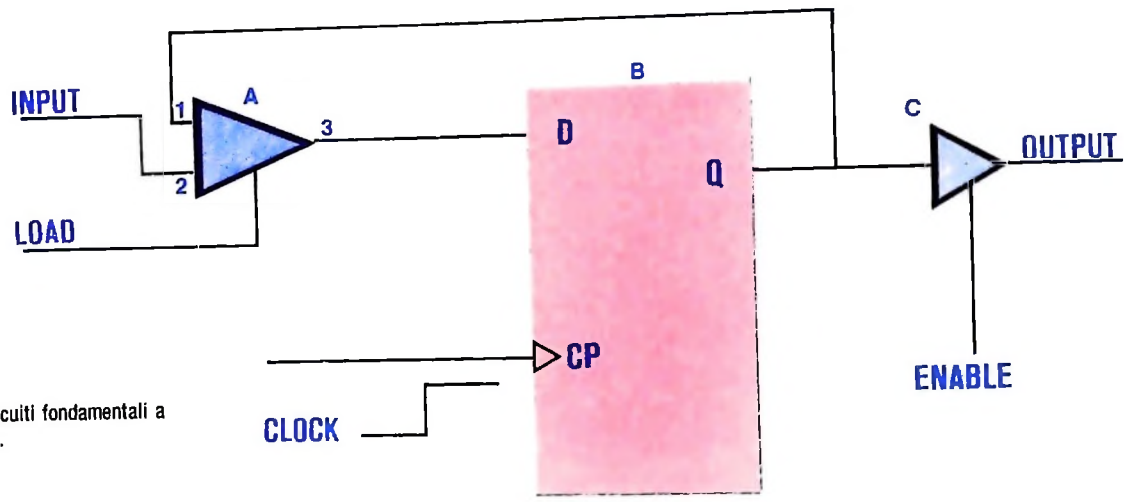


8



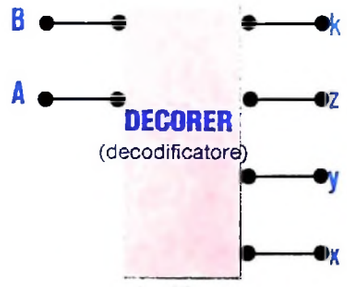
Due rappresentazioni grafiche dello stesso circuito BUFFER bidirezionale.

9



Uno dei circuiti fondamentali a three-state.

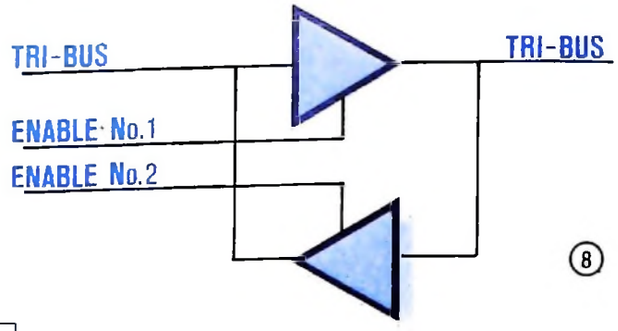
ria se non attraverso il comando WRITE. Quelli in lettura, invece, poiché vanno al BUS, devono passare prima attraverso i BUFFER a tre stati pilotati dal comando READ.
Ricapitolando tutto il processo, possiamo dire che, innanzitutto, devono essere stabili gli indirizzi; poi, se l'operazione è di scrittura, anche i dati devono essere stabili sulle linee di dati prima che arrivi il comando di WRITE. Se invece l'operazione è di lettura, i dati dalla memoria passano alle entrate dei BUFFER a tre stati e da qui al BUS con il comando di READ. La simbologia grafica dei BUFFER bidirezionali è visibile nella figura 8.
Vediamo infine il circuito fondamentale che ci permette di realizzare molti tipi di registri (figura 9). Possiamo considerarlo come il circuito sequenziale minimo per quantità di elementi. Prima di passare a spiegarne il funzionamento, vediamo in breve quello del circuito A. Esso si chiama MULTIPLEXER ed è un deviatore che, se riceve il comando LOAD = 0, smista all'uscita 3 il segnale presente all'ingresso 1. Se, invece, è LOAD = 1, è il segnale presente all'ingresso 2 a passare all'uscita 3.
Il circuito B è il classico flip-flop di tipo D, che conosciamo ormai bene, mentre il circuito C è un BUFFER a tre stati. Quando LOAD = 1, il dato presente nell'entrata 2 (INPUT) si presenta all'entrata D del flip-flop e passa all'uscita Q quando c'è una transizione positiva nel clock; quando LOAD = 0, l'uscita Q del flip-flop entra all'ingresso 1 del multiplexer e si ripresenta uguale all'uscita Q per ogni transizione positiva del clock. Il suo valore può cambiare solo quando LOAD = 1. Se ENABLE = 1, il valore di Q passa all'uscita, cioè al BUS a cui è collegato. In conclusione, quando abbiamo caricato un dato (zero o uno) nel flip-flop, questo dato rimane invariato anche se arrivano altre transizioni del clock. Possiamo passare esternamente questo dato (OUTPUT) attraverso il comando ENABLE. L'unico modo in cui possiamo cambiare l'informazione internamente è attraverso il comando LOAD e la transizione positiva del clock.



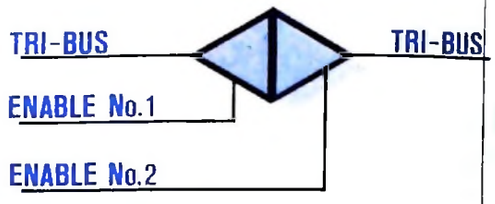
⑦

INPUT		OUTPUT			
A	B	x	y	z	k
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

Il decodificatore e la sua tabella di verità.

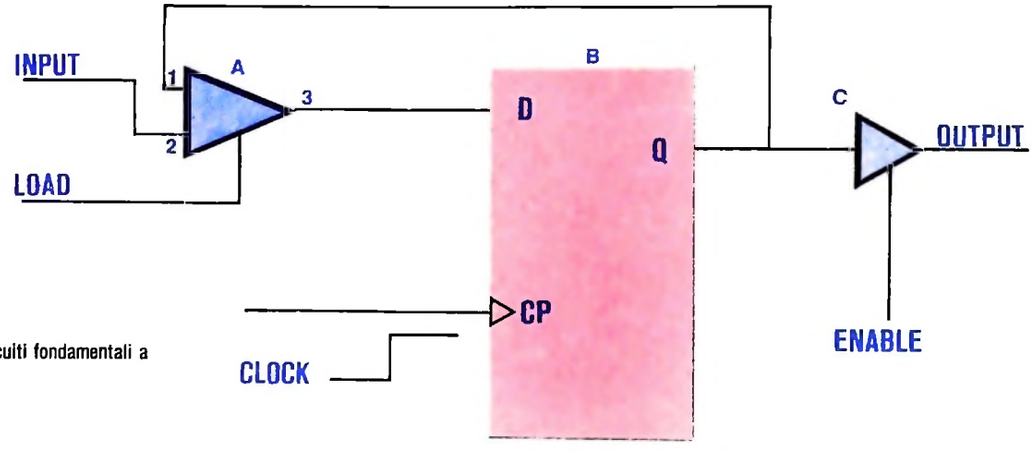


⑧



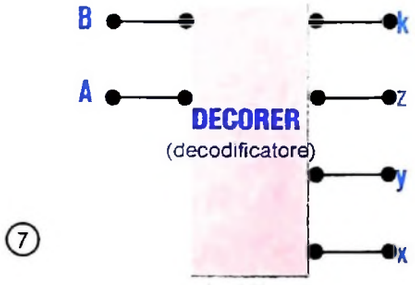
Due rappresentazioni grafiche dello stesso circuito BUFFER bidirezionale.

⑨



Uno dei circuiti fondamentali a three-state.

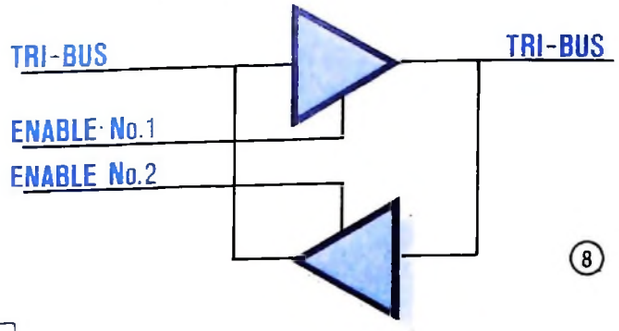
ria se non attraverso il comando WRITE. Quelli in lettura, invece, poiché vanno al BUS, devono passare prima attraverso i BUFFER a tre stati pilotati dal comando READ.
Ricapitolando tutto il processo, possiamo dire che, innanzitutto, devono essere stabili gli indirizzi; poi, se l'operazione è di scrittura, anche i dati devono essere stabili sulle linee di dati prima che arrivi il comando di WRITE. Se invece l'operazione è di lettura, i dati dalla memoria passano alle entrate dei BUFFER a tre stati e da qui al BUS con il comando di READ. La simbologia grafica dei BUFFER bidirezionali è visibile nella figura 8.
Vediamo infine il circuito fondamentale che ci permette di realizzare molti tipi di registri (figura 9). Possiamo considerarlo come il circuito sequenziale minimo per quantità di elementi. Prima di passare a spiegarne il funzionamento, vediamo in breve quello del circuito A. Esso si chiama MULTIPLEXER ed è un deviatore che, se riceve il comando LOAD = 0, smista all'uscita 3 il segnale presente all'ingresso 1. Se, invece, è LOAD = 1, è il segnale presente all'ingresso 2 a passare all'uscita 3.
Il circuito B è il classico flip-flop di tipo D, che conosciamo ormai bene, mentre il circuito C è un BUFFER a tre stati. Quando LOAD = 1, il dato presente nell'entrata 2 (INPUT) si presenta all'entrata D del flip-flop e passa all'uscita Q quando c'è una transizione positiva nel clock; quando LOAD = 0, l'uscita Q del flip-flop entra all'ingresso 1 del multiplexer e si ripresenta uguale all'uscita Q per ogni transizione positiva del clock. Il suo valore può cambiare solo quando LOAD = 1. Se ENABLE = 1, il valore di Q passa all'uscita, cioè al BUS a cui è collegato. In conclusione, quando abbiamo caricato un dato (zero o uno) nel flip-flop, questo dato rimane invariato anche se arrivano altre transizioni del clock. Possiamo passare esternamente questo dato (OUTPUT) attraverso il comando ENABLE. L'unico modo in cui possiamo cambiare l'informazione internamente è attraverso il comando LOAD e la transizione positiva del clock.



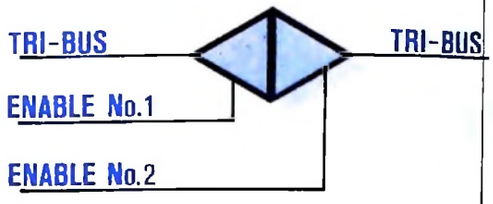
⑦

INPUT		OUTPUT			
A	B	x	y	z	k
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

Il decodificatore e la sua tabella di verità.

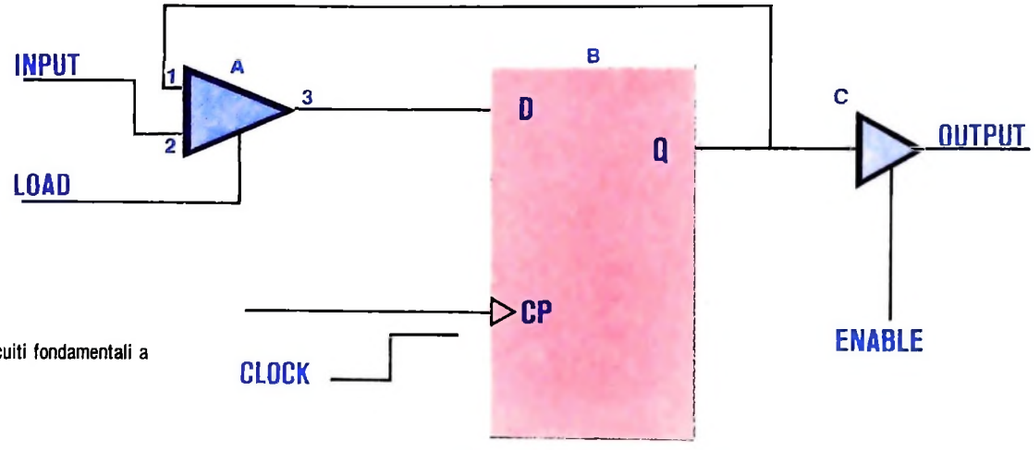


⑧



Due rappresentazioni grafiche dello stesso circuito BUFFER bidirezionale.

⑨



Uno dei circuiti fondamentali a three-state.

ria se non attraverso il comando WRITE. Quelli in lettura, invece, poiché vanno al BUS, devono passare prima attraverso i BUFFER a tre stati pilotati dal comando READ.
Ricapitolando tutto il processo, possiamo dire che, innanzitutto, devono essere stabili gli indirizzi; poi, se l'operazione è di scrittura, anche i dati devono essere stabili sulle linee di dati prima che arrivi il comando di WRITE. Se invece l'operazione è di lettura, i dati dalla memoria passano alle entrate dei BUFFER a tre stati e da qui al BUS con il comando di READ. La simbologia grafica dei BUFFER bidirezionali è visibile nella figura 8.
Vediamo infine il circuito fondamentale che ci permette di realizzare molti tipi di registri (figura 9). Possiamo considerarlo come il circuito sequenziale minimo per quantità di elementi. Prima di passare a spiegarne il funzionamento, vediamo in breve quello del circuito A. Esso si chiama MULTIPLEXER ed è un deviatore che, se riceve il comando LOAD = 0, smista all'uscita 3 il segnale presente all'ingresso 1. Se, invece, è LOAD = 1, è il segnale presente all'ingresso 2 a passare all'uscita 3.
Il circuito B è il classico flip-flop di tipo D, che conosciamo ormai bene, mentre il circuito C è un BUFFER a tre stati. Quando LOAD = 1, il dato presente nell'entrata 2 (INPUT) si presenta all'entrata D del flip-flop e passa all'uscita Q quando c'è una transizione positiva nel clock; quando LOAD = 0, l'uscita Q del flip-flop entra all'ingresso 1 del multiplexer e si ripresenta uguale all'uscita Q per ogni transizione positiva del clock. Il suo valore può cambiare solo quando LOAD = 1. Se ENABLE = 1, il valore di Q passa all'uscita, cioè al BUS a cui è collegato. In conclusione, quando abbiamo caricato un dato (zero o uno) nel flip-flop, questo dato rimane invariato anche se arrivano altre transizioni del clock. Possiamo passare esternamente questo dato (OUTPUT) attraverso il comando ENABLE. L'unico modo in cui possiamo cambiare l'informazione internamente è attraverso il comando LOAD e la transizione positiva del clock.

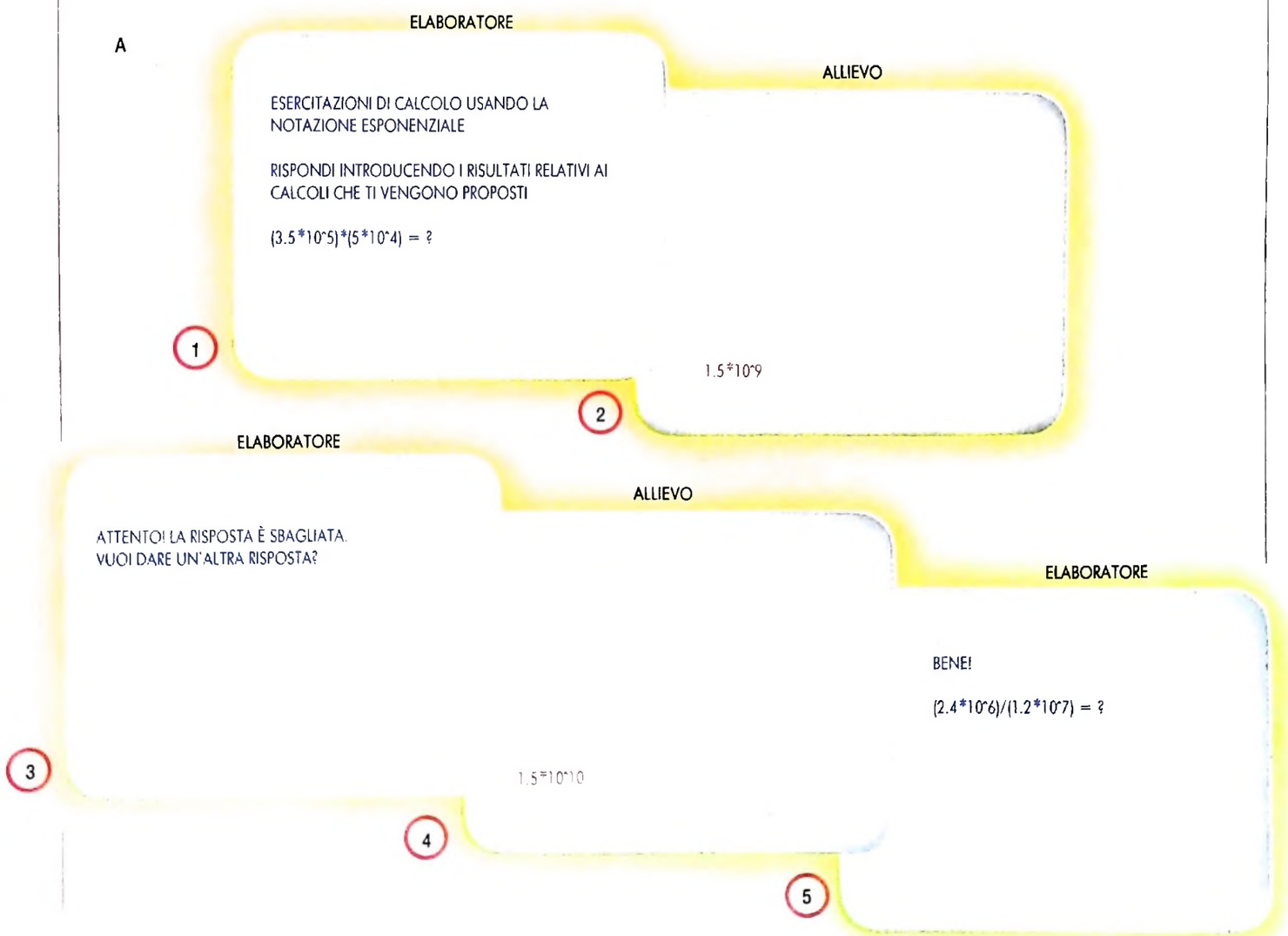
LE ESERCITAZIONI

Un'altra metodologia dell'informatica applicata alla didattica.

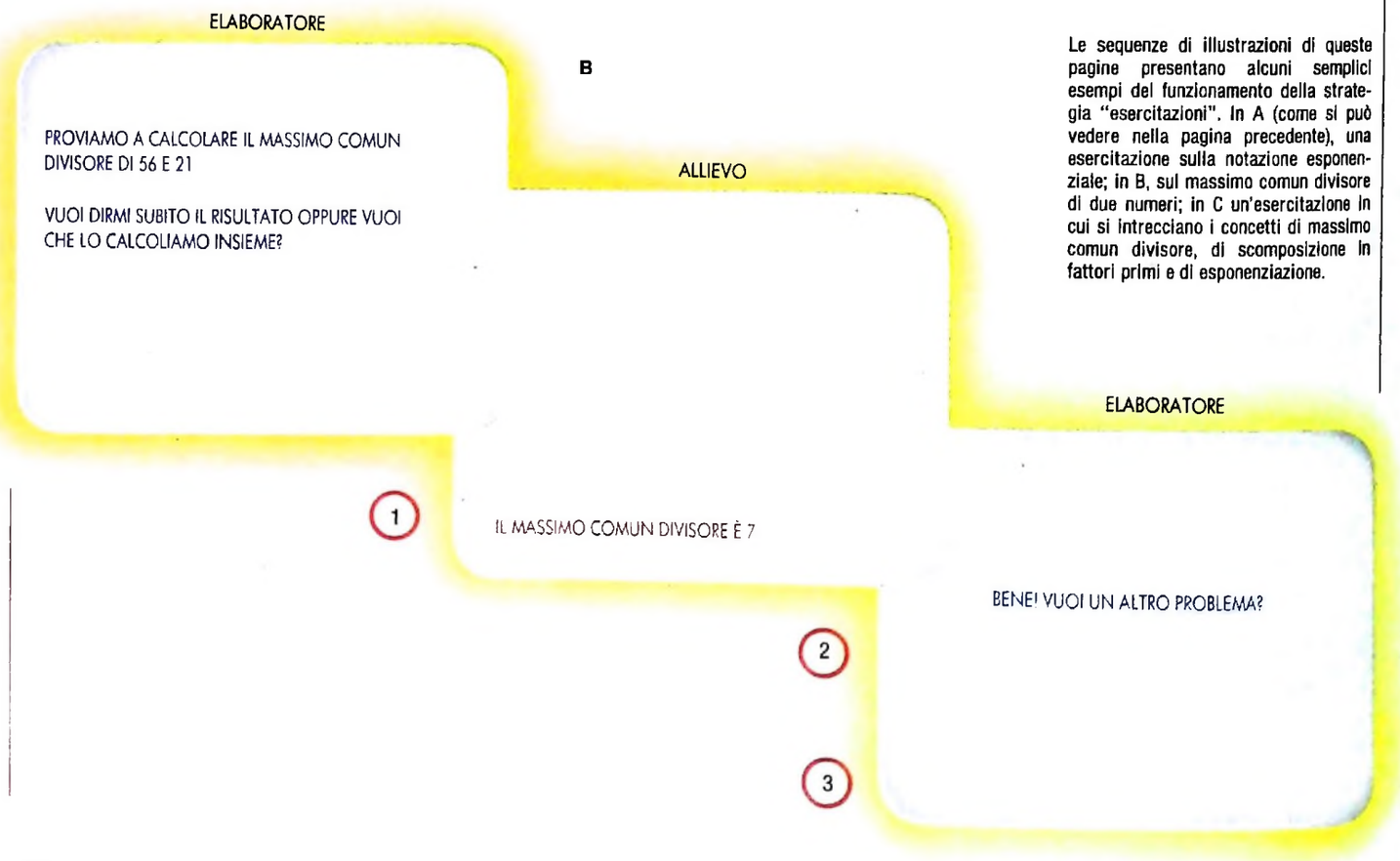
Continuiamo nella presentazione e nell'analisi delle metodologie impiegate nelle applicazioni dell'informatica alla didattica, esaminando la metodologia delle *esercitazioni* (in inglese "drill and practice").
Anche in questo caso, come in tutte le altre strategie usate, i livelli sono diversi e si passa da un sistema estremamente semplice ad altri molto più complessi e articolati.
Nel caso più semplice il sistema presenta all'allievo una serie di esercizi su un dato argomento. L'allievo deve limitarsi a risolverli e a dare le risposte richieste.
In molti casi l'elaboratore si limita a segnalare allo studente se la risposta è corretta oppure sbagliata. Nel caso di risposta sbagliata ripropone semplicemente l'esercizio una, due o tre volte e, alla fine, dopo un certo numero di errori, visualizza la risposta corretta e passa ad un esercizio successivo.

Miglioramenti del sistema

Un primo arricchimento del sistema consiste nel prendere nota del numero di esercizi che hanno ricevuto risposte corrette, mettendoli in rapporto con il totale degli esercizi, e for-



nire pertanto una valutazione generale della prova stessa.
Questo è un primo passo verso l'impiego dell'elaboratore come strumento originale nella valutazione e soprattutto nell'autovalutazione, su cui torneremo prossimamente.
Un arricchimento sostanziale si ha invece quando la soluzione degli esercizi, a richiesta dello studente prima della sua risposta, o comunque dopo una prima risposta sbagliata, viene guidata dall'elaboratore.
Negli esempi si vede prima il caso in cui lo studente risponde subito con il risultato corretto, poi quello in cui il sistema interviene in suo aiuto.
Come si può notare la metodologia si è arricchita di aspetti relativi alla strategia di tipo "tutoriale", della quale si è parlato precedentemente.
L'aspetto più interessante è l'interattività: l'allievo è chiamato continuamente a dare risposte attive alla macchina e a costruire a poco a poco la soluzione dell'esercizio.
A questo punto però resta aperto tutto un vasto campo di applicazioni che trascendono la pura metodologia delle "esercitazioni": del resto un sistema ricco è appunto un sistema che usa diversi tipi di metodologie in interazione reciproca, sfruttando al meglio i vantaggi che ciascuna di esse può offrire.
Tornando alla semplice strategia delle esercitazioni, un altro

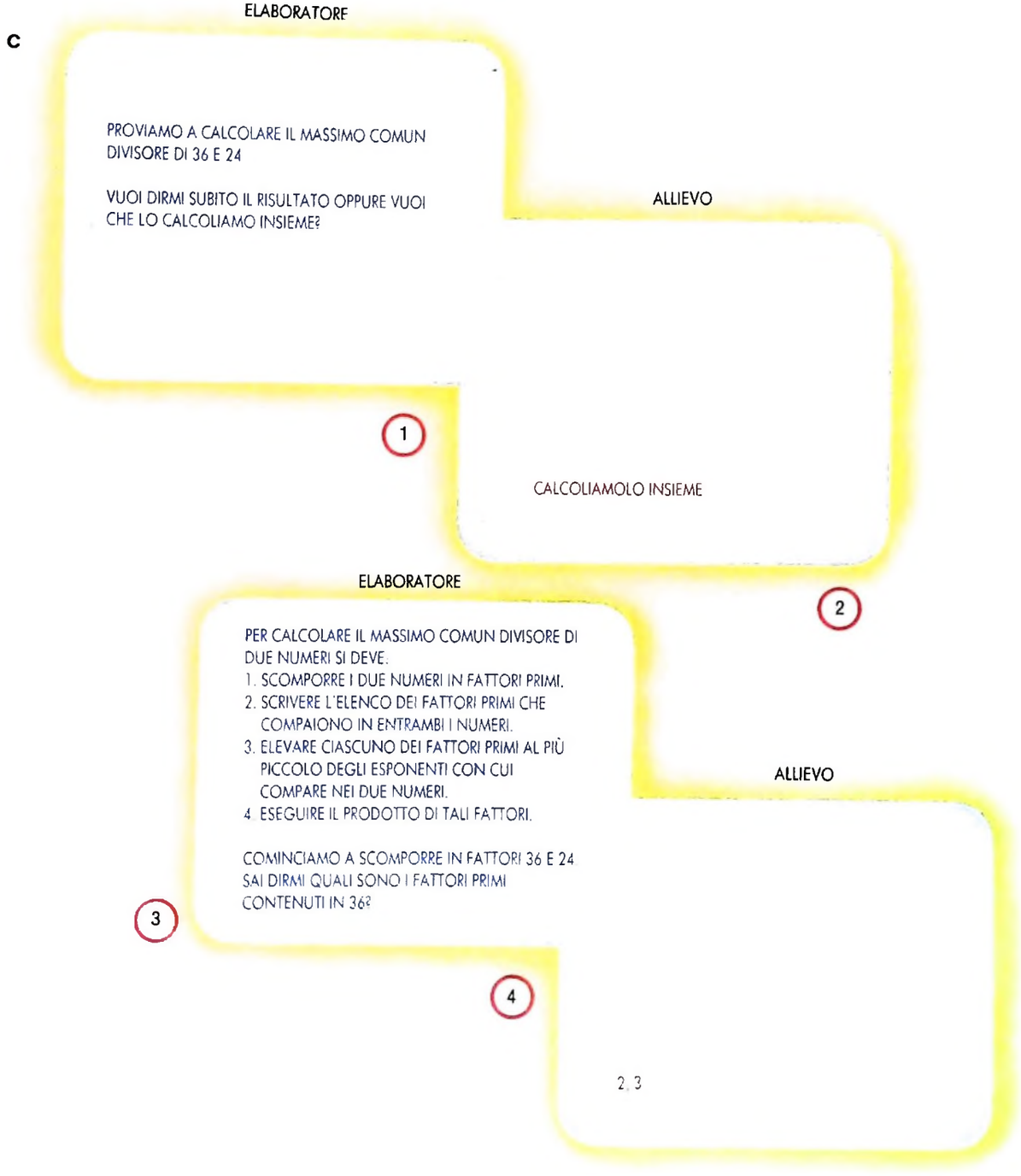


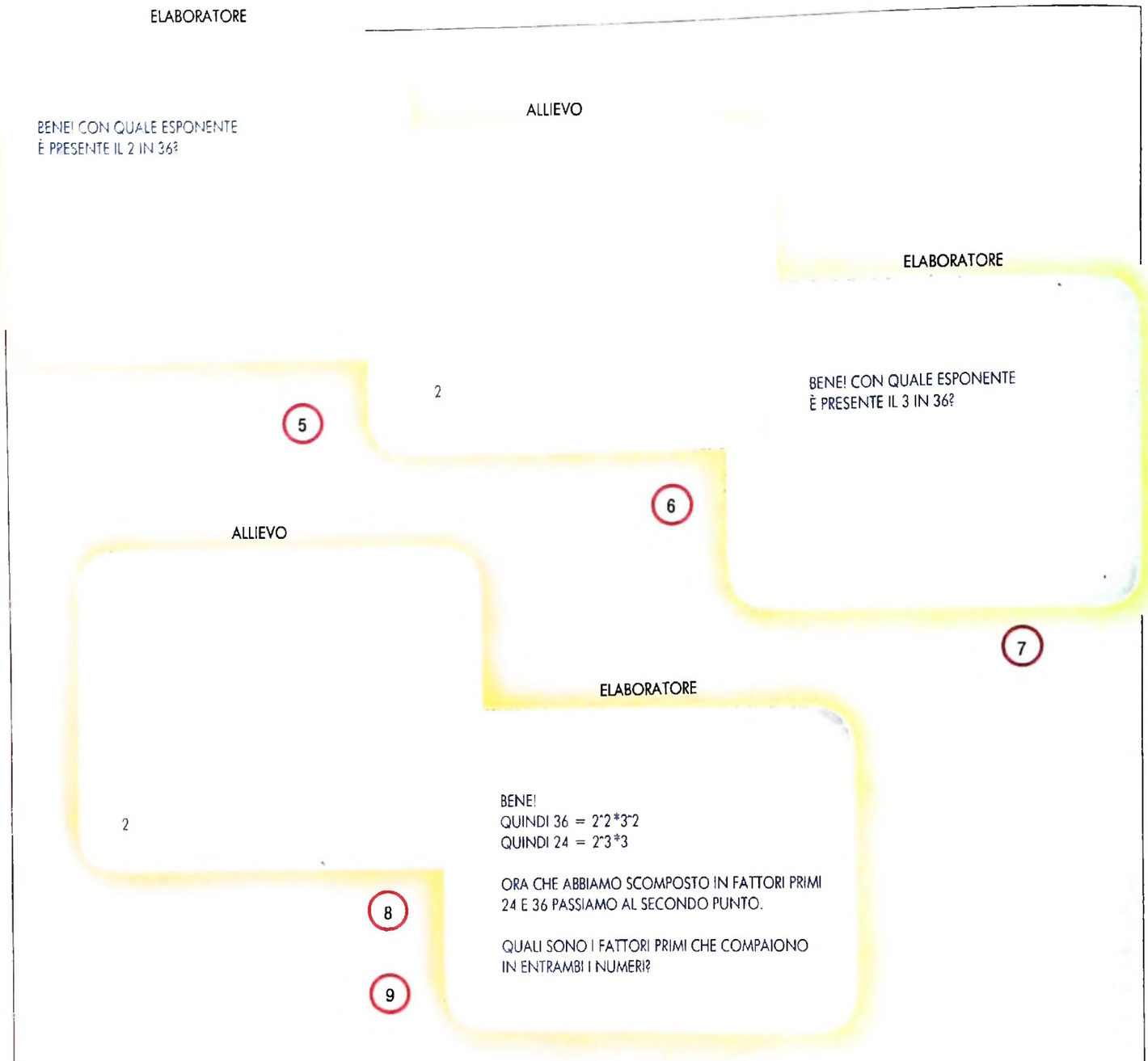
Le sequenze di illustrazioni di queste pagine presentano alcuni semplici esempi del funzionamento della strategia "esercitazioni". In A (come si può vedere nella pagina precedente), una esercitazione sulla notazione esponenziale; in B, sul massimo comun divisore di due numeri; in C un'esercitazione in cui si intrecciano i concetti di massimo comun divisore, di scomposizione in fattori primi e di esponenziazione.

miglioramento è la graduazione degli esercizi con difficoltà crescenti a seconda delle risposte dell'allievo.
Si comincia con esercizi semplici; se vengono superati si passa gradualmente a quelli più complessi, altrimenti si ripropongono altri esercizi semplici.
La graduazione delle difficoltà è un compito molto impegnativo per chi deve preparare il programma: infatti un programma ben strutturato deve poter offrire un nutrito ventaglio di difficoltà ai tipi di studenti più disparati e soprattutto valutare il grado di dimestichezza con un certo tipo di esercizio per poter passare al tipo con difficoltà maggiore.

Esercizi identici, con dati diversi

Ancora un miglioramento: l'elaboratore può presentare un numero indefinito di esercizi dello stesso tipo, semplicemente cambiando i dati di uno stesso problema.
Facciamo un esempio diretto, mediante un problema elementare di cinematica.
"Un'automobile parte da una località e viaggia ad una velocità costante V1. Dopo un certo tempo t, parte dalla stessa località e nella stessa direzione un'altra macchina, con la velocità V2, pure costante. Dopo quanto tempo e dopo quanti





chilometri la seconda macchina raggiungerà la prima?"
Il programma assegna alle variabili V1, V2 e t dei valori pseudocasuali, entro ambiti stabiliti e nel rispetto delle condizioni richieste dal particolare problema (ad esempio, nel nostro caso, deve valere V2>V1).
L'elaboratore, se questo entra nella strategia generale del programma, può ripresentare un numero indeterminato di questi problemi con dati numerici diversi, fino a che l'allievo abbia acquistato una sufficiente sicurezza.

Vantaggi e svantaggi

Concludendo possiamo dire che il vantaggio della metodologia "esercitazioni" è essenzialmente quello di liberare l'insegnante dal compito puramente ripetitivo di proporre e risolvere esercizi, soprattutto gli esercizi di tipo standard.
L'elaboratore può mettere a disposizione di tanti studenti tutte le proprie capacità che nel caso dei programmi più semplici sono anzitutto ripetitive, mentre nei sistemi più complessi sono anche di aiuto e di guida alla risoluzione.
Uno svantaggio è dato dal fatto che, applicando la metodologia nella sua forma più pura, si ottiene soltanto un addestramento e manca una vera e propria concettualizzazione.
È evidente però che da un lato, come si è detto, nulla vieta di integrare questa con altre strategie, e dall'altro che anche il compito addestrativo non va sottovalutato, a fianco degli altri compiti della didattica.
Quindi se l'elaboratore è in grado di assolvere il compito di addestratore, sia il benvenuto.

Lezione 13

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

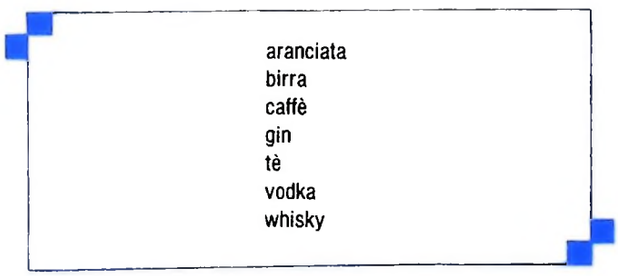
MEDST. DO
MEDSP. BA
contenuti nella cassetta "5 esercizi di programmazione", lato A.

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

Abbiamo esaminato l'algoritmo di ricerca sequenziale in un array: tale algoritmo è evidentemente molto semplice, ma rischia di diventare molto poco efficiente quando ci si trovi in presenza di una grande quantità di informazioni tra cui effettuare la ricerca; infatti, abbiamo visto come, in media, ci si debba aspettare di dover scandire la metà degli elementi dell'array; inoltre, la velocità di reperimento di un'informazione dipende fortemente dalla posizione dell'informazione cercata nell'array.
Pertanto, si ricorre normalmente ad algoritmi differenti, che risolvano tali problemi: tra questi, uno dei più semplici è il cosiddetto algoritmo di ricerca "dicotomica" o "binaria". Requisito di tale algoritmo è che i dati in tabella siano ordinati.
Il modello di comportamento che seguiamo è analogo a quello della ricerca di un nominativo nell'elenco telefonico. Infatti, se per esempio cerchiamo il numero di telefono di Rossi Mario operiamo come segue:

- apriamo a metà l'elenco;
- SE la pagina è quella in cui compare Rossi
  - ALLORA cerchiamo il numero di Rossi Mario
  - ALTRIMENTI
    - SE la pagina è su nomi "minori" (in senso alfabetico) di Rossi
      - ALLORA ripetiamo il procedimento sulle pagine di elenco che precedono la pagina aperta;
      - ALTRIMENTI ripetiamo il procedimento sulle pagine di elenco che seguono la pagina aperta.

Supponiamo, a titolo di esempio, di effettuare una ricerca nella seguente tabella, in cui i dati sono ordinati alfabeticamente:



e di voler indicare, per esempio, se la parola "caffè" è o meno presente. La ricerca prosegue dunque nel modo seguente:

- si individua l'elemento mediano dell'array calcolando il valore dell'indice relativo come valore medio tra l'indice superiore e inferiore dell'array (M:=I+Y/2 dove I e Y sono rispettivamente l'indice inferiore e superiore dell'array, in questo caso rispettivamente 0 e 6. Ci interessa che M sia solo la parte intera della divisione, poi-

ché non avrebbe senso assegnare all'indice di un array un valore decimale).

• Si verifica quindi se l'elemento selezionato è uguale all'elemento cercato. Nel nostro caso, l'operazione di selezione dell'elemento mediano fornisce il valore 3 (gli indici inferiore e superiore hanno infatti rispettivamente i valori 0 e 6), facendoci quindi "puntare" all'elemento che contiene l'informazione "gin".

A questo punto, se l'elemento selezionato è quello cercato la ricerca ha termine; altrimenti si verifica se l'elemento cercato è maggiore o minore.

Se è maggiore allora la ricerca viene ripetuta sulla seconda metà dell'array, scartando la prima che evidentemente contiene solo elementi minori. Se invece è minore si seleziona la prima metà dell'array.

Nel nostro caso, poiché "caffè" è minore di "gin", verrà selezionata la prima metà, che contiene gli elementi compresi tra "aranciata" e "gin". Tuttavia, poiché quest'ultimo è già stato esaminato, lo scarteremo: il nuovo array su cui effettuare la ricerca sarà compreso tra la posizione 0 (iniziale) e 2 (immediatamente precedente a "gin"). Si tratta ora di ricalcolare la posizione media della parte di array selezionata. Nel nostro caso la nuova posizione da verificare è quella il cui indice è la media tra 0 (posizione di "aranciata") e 2 (posizione subito prima di "gin"), cioè 1 che indica quindi l'elemento contenente l'informazione "birra".

Ripetiamo il confronto e, poiché l'elemento individuato non è quello ricercato, selezioniamo la successiva porzione di array su cui ripetere la ricerca: sarà, in questo caso, la seconda parte della tabella, che va da "caffè" (l'elemento successivo a "birra") a "caffè" (che era l'estremo già selezionato). A questo punto la ricerca si conclude, in quanto la nostra tabella si è ridotta a un solo elemento che nel nostro caso è proprio quello richiesto.

Esaminiamo dunque lo schema del programma, scritto come al solito in PASCAL (supponiamo che la suddetta tabella sia già stata riempita): anzi non limitiamo il problema alla semplice individuazione della presenza o meno dell'informazione nella tabella, ma tornando all'esempio del listino prezzi, facciamo in modo che visualizzi il prezzo relativo:

- Chiede consumazione da cercare
- Pone I (indice inferiore) = 0
- Pone Y (indice superiore) = 6
- Pone indicatore di trovato = "no"
- REPEAT
  - Pone M (posizione media dell'array selezionato) = I+Y/2
  - IF elemento (M) = consumazione cercata
    - THEN pone indicatore trovato = "sì"
    - ELSE
      - IF elemento (M) > consumazione cercata
        - THEN pone Y=M—1
        - ELSE
          - Pone I=M+1

- UNTIL trovato = "si" OR I > Y
- IF trovato = "no" THEN
  - Informa che l'elemento non è presente
  - ELSE
    - Informa che l'elemento è presente e ne fornisce il prezzo

Vediamo alcune osservazioni sull'algoritmo mostrato. Si noti innanzitutto come ad ogni successiva passata dopo la prima, la selezione della porzione di array su cui effettuare la ricerca avvenga aggiornando il valore dell'indice medio diminuito o aumentato di 1 per scartare l'elemento che è già stato analizzato.
In questo modo, i due indici si vanno via via avvicinando: la ricerca si interrompe o quando l'elemento cercato viene trovato o quando i successivi aggiornamenti degli indici li portano a sovrapporsi (I diventa maggiore di Y) e conseguentemente siamo certi che l'elemento cercato non è presente avendo esaurito gli elementi di array su cui continuare la ricerca.
Vediamo quindi il programma BASIC che realizza la ricerca:

```
10 DIM C$(6),P(6)
20 REM Lettura dati
30 FOR I=0 TO 6
40 READ C$(I),P(I)
50 NEXT I
60 REM Richiesta consumazione da ricercare
70 INPUT "Consumazione";A$
80 REM Inizio ricerca
90 LET I=0
100 LET Y=6
110 LET M=(I+Y)/2
120 IF C$(M)=A$ THEN 200
130 IF C$(M)<A$ THEN LET I=M+1 ELSE LET Y=M-1
150 IF INT(I)<=INT(Y) THEN 110
160 PRINT "Consumazione inesistente"
170 GOTO 220
200 PRINT "Prezzo:";P(M)
220 REM Fine
1000 DATA "ARANCIATA",1000,"BIRRA",1000,"CAFF
E'",500,"GIN",1600
1010 DATA "THE",800,"VODKA",1600,"WHISKY",1800
```

Si noti la linea 150 dove si controlla la condizione di fine ricerca, confrontando la parte intera dei due indici I e Y ottenuta con la funzione INT; se ciò non fosse fatto, si rischierebbe di interrompere la ricerca perdendo uno degli elementi della porzione di array ancora da esaminare: risulterebbero così assenti elementi che in realtà non lo sono.
Per meglio comprendere il comportamento dell'algoritmo, è consigliabile comportarsi come se fossimo l'esecutore dell'algoritmo: disegniamo su un foglio di carta alcune "scatole" che simuleranno le variabili e partendo dalla prima istruzione ese-

guiamo il programma passo per passo, eseguendo i comandi che ci vengono via via impartiti. Quando troviamo un'istruzione di assegnamento, cancelliamo il precedente valore della "scatola-variabile" corrispondente e scriviamo quello nuovo.
Questa tecnica di simulazione è in generale molto utile per la comprensione di programmi, ma ha anche una funzione molto importante nell'attività cosiddetta di "messa a punto" dei programmi in quanto permette di individuare da quale punto dell'esecuzione si crea una condizione d'errore.

Alcune osservazioni sulle prestazioni della ricerca dicotomica

Abbiamo visto che su una tabella di 1000 elementi una ricerca sequenziale richiede mediamente 500 accessi per il reperimento di un'informazione.
Proviamo ora a esaminare il numero di accessi necessari nel caso di una ricerca dicotomica.
Supponiamo di effettuare la ricerca su una tabella di 128 elementi ed elenchiamo tutti i possibili successivi accessi che faremmo se l'informazione non fosse presente o comunque venisse reperita all'ultima passata:

1ª passata: passaggio da una tabella di 128 elementi a una di 64;
2ª passata: passaggio da una tabella di 64 elementi a una di 32;
3ª passata: passaggio da una tabella di 32 elementi a una di 16;
4ª passata: passaggio da una tabella di 16 elementi a una di 8;
5ª passata: passaggio da una tabella di 8 elementi a una di 4;
6ª passata: passaggio da una tabella di 4 elementi a una di 2;
7ª passata: passaggio da una tabella di 2 elementi a una di 1.

Cioè, con 7 passate effettuiamo una ricerca su una tabella di 128 elementi. Se la tabella fosse grande il doppio, avremmo bisogno di una sola passata in più (riduzione da 256 a 128 elementi) e così via.
Il matematico ci dice che il numero di passate è N, in modo che

$$2\hat{}N = \text{dimensione della tabella}$$

(infatti $2\hat{}7 = 128$).
Il matematico sempre dice che N è il logaritmo in base 2 di 128. Per questo motivo la ricerca dicotomica viene anche detta "logaritmica".

Cosa abbiamo imparato

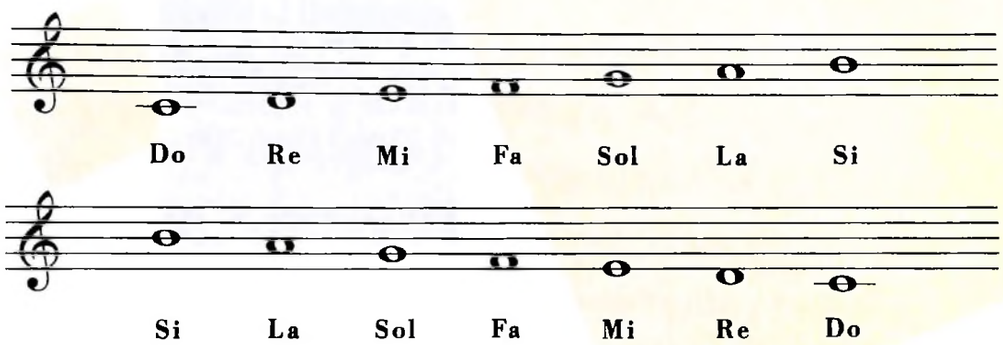
In questa lezione abbiamo imparato:

- l'algoritmo di ricerca dicotomica;
- la realizzazione BASIC di un programma che effettua una ricerca dicotomica;
- le prestazioni dell'algoritmo di ricerca dicotomica confrontate con quelle della ricerca sequenziale.

IL MODO MAGGIORE

Come realizzare la trasposizione di un brano musicale mediante un programma.

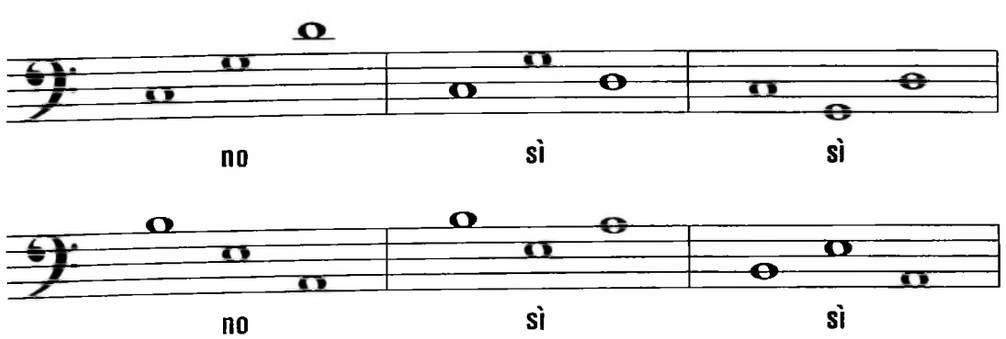
I modi musicali sono tipicamente ascendenti o discendenti. Il modo maggiore, come abbiamo già accennato, è costituito dalle altezze della scala diatonica; il modo maggiore ascendente ed il modo maggiore discendente sono quindi costituiti dalle seguenti successioni di altezze:



Relazioni tra i gradi del modo maggiore

I gradi più *importanti* del modo maggiore sono quelli corrispondenti all'innalzamento o all'abbassamento di un intervallo di quinta della tonica e cioè la dominante (quinto grado della scala: V) e la sottodominante (quarto grado della scala: IV); questi gradi hanno un ruolo più rilevante proprio perché hanno questa relazione di quinta (ascendente o discendente) con la tonica. Nel costruire sequenze musicali in una certa tonalità maggiore dobbiamo tener presenti almeno due indicazioni elementari:

a) evitare due intervalli melodici di quarta o di quinta nella stessa direzione, poiché i suoni iniziale e finale formano una dissonanza: nel primo caso le due quarte formano un inter-



vallo di 10 semitoni (cioè un intervallo di settima minore) e nel secondo caso le due quinte formano un intervallo di 14 semitoni (cioè un intervallo di nona maggiore); in generale possiamo considerare le sequenze che creano dissonanze come non melodiche, e quindi da evitare;

b) non possiamo utilizzare *salti* (cioè intervalli) più ampi di una quinta in una melodia (eccezion fatta per l'intervallo d'ottava).

Queste due regole hanno carattere meramente indicativo e fanno riferimento alla prassi della musica tonale; esemplifichiamo la prima regola con le successioni di intervalli rappresentate nella pagina precedente, in basso.

gramma qui sotto. Esso genera sequenze pseudocasuali di altezze di una certa tonalità maggiore che noi selezioniamo indicando con 1 il do maggiore, con 12 il do diesis maggiore, con 11 il re maggiore e così decrescendo fino al si maggiore con 2: indicando una certa tonalità piuttosto che un'altra (a condizione di mantenere identiche le altre risposte), trasponiamo tutte le altezze generate di un numero ben preciso di semitoni. Come negli esempi precedenti, si tratta di un programma senza terminazione che produce sequenze diverse in dipendenza dal numero intero che gli diamo, in base al quale calcola il primo numero casuale. Per quanto riguarda le diverse fasi del programma, le istruzioni 025-090 generano i

```
010 clear
011 REM scelta della tonalita'
maggiore
012 input "tonalita' (tra 1 e 12
)";xt
013 REM scelta del tempo
014 input "tempo (tra 0 e 6)";tm
015 t=2^tm
015 REM generazione dei numeri
pseudocasuali
017 input "dammi un numero inte
ro";INIT
020 rn=rnd(INIT)
025 REM generazione dei gradi
della scala maggiore
030 if rn>0 and rn<=0.1 then x=4697
040 if rn>0.1 and rn<=0.2 then
x=4184
050 if rn>0.2 and rn<=0.3 then
x=3728
060 if rn>0.3 and rn<=0.4 then
x=3516
070 if rn>0.4 and rn<=0.5 then
x=3134
080 if rn>0.5 and rn<=0.6 then
x=2793
090 if rn>0.6 and rn<=0.7 then
x=2484
095 REM generazione di pause
100 if rn>0.7 and rn<=0.8 then x=0
105 REM salto di un'ottava sopra
110 if rn>0.8 and rn<=0.9 then
x=x/2
115 REM salto di un'ottava sotto
120 if rn>0.9 and rn<=1 then x=x*2
140 REM generazione della tonalita'
141 if xt=2 then x=x*1.059463
142 if xt=3 then x=x*1.122462
143 if xt=4 then x=x*1.189207
144 if xt=5 then x=x*1.259921
145 if xt=6 then x=x*1.334840
146 if xt=7 then x=x*1.414214
147 if xt=8 then x=x*1.498307
148 if xt=9 then x=x*1.597401
149 if xt=10 then x=x*1.681793
150 if xt=11 then x=x*1.781797
151 if xt=12 then x=x*1.887749
152 REM generazione delle durate
153 y=rn*255/t
155 REM generazione delle note e
controlli
156 if x>16383 then x=x/2
157 if y>255 then y=y/2
160 sound x,y
170 goto 020
```

Le tonalità maggiori e la trasposizione

Abbiamo visto che la tonalità maggiore è caratterizzata dalla sequenza di intervalli ascendenti 2-2-1-2-2-2-1 (espressi in semitoni); questo significa che possiamo definire 12 differenti tonalità maggiori applicando la sequenza di intervalli del modo maggiore ad ognuno dei 12 semitoni della scala temperata. È semplice realizzare la trasposizione di un brano musicale mediante un programma: usiamo una codifica delle altezze riferita ad una particolare tonica tenendo fissa la struttura di intervalli del modo maggiore; un esempio è il programma gradi della scala maggiore; le istruzioni 140-151 generano la tonalità che abbiamo prescelto moltiplicando i gradi della scala di base per l'opportuna potenza della radice dodicesima di due (cioè abbassando di un certo numero di semitoni). Inoltre, l'istruzione 100 genera pause, l'istruzione 110 innalza di un'ottava, l'istruzione 120 abbassa di un'ottava, l'istruzione 153 genera il valore di durata sulla base del numero casuale attuale e dell'indicazione di tempo che abbiamo dato (tra 0 e 6, dal lento al veloce). Con tale impostazione possiamo cambiare tonalità semplicemente cambiando la particolare tonica a cui tutto il testo musicale si riferisce.

Esercizio

Possiamo codificare l'ouverture delle *Nozze di Figaro* di Mozart in un'infinità di modi differenti; il più semplice e più vicino alla partitura è descrivere la sequenza di note con istruzioni SOUND che generano valori fissati di frequenze corrispondenti alle note e che utilizzano valori di durata espliciti in rapporto ad una indicazione di tempo; il programma che segue realizza questo tipo di approccio:

```
010 clear
020 DQ5=4184
030 C5D=4433
040 P=0
050 E5=3728
060 F5D=3321
070 GQ5=3134
080 AQ5=2793
090 G5D=2954
100 A5D=2636
110 B5=2484
120 D5D=3950
130 A4=5586
140 B4=4968
143 INPUT "tempo (da 0 a 6)";tm
146 Y=512/2^tm
150 REM NOZZE DI FIGARO (ouverture)
159 REM battuta 1
160 SOUND DQ5,Y/8
170 SOUND C5D,Y/8
180 SOUND DQ5,Y/8
190 SOUND C5D,Y/8
200 SOUND DQ5,Y/4
210 SOUND P,Y/4
219 REM battuta 2
220 SOUND DQ5,Y/8
230 SOUND C5D,Y/8
240 SOUND DQ5,Y/8
250 SOUND E5,Y/8
260 SOUND F5D,Y/8
270 SOUND E5,Y/8
280 SOUND F5D,Y/8
290 SOUND GQ5,Y/8
299 REM battuta 3
300 SOUND AQ5,Y/8
310 SOUND G5D,Y/8
320 SOUND AQ5,Y/8
330 SOUND G5D,Y/8
340 SOUND AQ5,Y/4
350 SOUND P,Y/4
359 REM battuta 4
360 SOUND AQ5,Y/8
370 SOUND G5D,Y/8
380 SOUND AQ5,Y/8
390 SOUND A5D,Y/8
400 SOUND B5,Y/8
410 SOUND AQ5,Y/8
420 SOUND GQ5,Y/8
430 SOUND F5D,Y/8
439 REM battuta 5
440 SOUND E5,Y/8
450 SOUND D5D,Y/8
460 SOUND E5,Y/8
470 SOUND F5D,Y/8
480 SOUND GQ5,Y/8
490 SOUND F5D,Y/8
500 SOUND E5,Y/8
510 SOUND DQ5,Y/8
519 REM battuta 6
520 SOUND C5D,Y/8
530 SOUND DQ5,Y/8
540 SOUND E5,Y/8
550 SOUND DQ5,Y/8
560 SOUND C5D,Y/8
570 SOUND A4,Y/8
580 SOUND B4,Y/8
590 SOUND C5D,Y/8
599 REM fine
600 SOUND DQ5,Y/4
610 END
```

Un altro criterio che possiamo seguire è di codificare la sequenza di intervalli del brano piuttosto che i valori di altezze e analogamente i rapporti tra la durata delle note piuttosto che i valori espliciti di durata; in questo modo è necessario indicare esplicitamente solo la prima nota con il suo valore di altezza e il suo valore di durata; la sequenza di intervalli è data dalla seguente sequenza di aumentazioni o diminuzioni per semitoni; per brevità omettiamo la codifica del programma corrispondente poiché è veramente banale.

−1 +1 −1 +1 pausa | 0 −1 +1 +2 +2 −2 +2 +1 |
+2 −1 +1 −1 +1 pausa | 0 −1 +1 +1 +1 −2 −2 −1 |
−2 −1 +1 +2 +1 −1 −2 −2 | −1 +1 +2 −2 −1 −4 +2 +2 | +1

Un terzo criterio è di descrivere parametricamente la tonalità così da poter scegliere di volta in volta la tonalità in cui ascoltare l'ouverture; è sufficiente che al posto delle istruzioni iniziali (10-150) che abbiamo visto nel programma precedente sostituiamo le istruzioni iniziali del seguente (10-73):
In questo modo, se diamo il valore 4184 otteniamo la versione originale; con qualunque altro valore trasponiamo il brano intero in alto (numeri più piccoli) o in basso (numeri più grandi).

```
010 clear
020 input "tempo (da 0 a 6)";tm
030 Y=512/2^tm
040 REM NOZZE DI FIGARO (ouverture)
050 REM calcolo altezze
055 PRINT "la tonalita' originale
    si ottiene con il valore 4184;"
060 input "prima altezza
    (da 128 a 8192)";pra
061 DQ5=pra
062 C5D=pra*1.059463
063 P=0
064 E5=pra/1.122462
065 F5D=pra/1.259921
066 G05=pra/1.334840
067 AQ5=pra/1.498307
068 G5D=pra/1.414212
069 A5D=pra/1.597401
070 B5=pra/1.681793
071 D5D=pra/1.059463
072 A4=pra*1.334840
073 B4=pra*1.189207
159 REM battuta 1
160 SOUND DQ5,Y/8
170 SOUND C5D,Y/8
180 SOUND DQ5,Y/8
190 SOUND C5D,Y/8
200 SOUND DQ5,Y/4
210 SOUND P,Y/4
219 REM battuta 2
220 SOUND DQ5,Y/8
230 SOUND C5D,Y/8
      .
      .
      .
      .
```

Osservando, infine, la presenza di strutture ritmiche e melodiche che si ripetono identiche o trasposte possiamo utilizzare il concetto di **subroutine** e, ad esempio, scrivere un programma in cui la battuta 1 e la battuta 3 sono generate mediante la seguente codifica, da inserire nel programma al posto della vecchia descrizione sequenziale (istruzioni 159-210 la prima e istruzioni 299-350 la terza).

```
159 REM battuta 1
160 op=1
170 gosub 1000
  .
  .

299 REM battuta 3
300 op=1.498307
310 gosub 1000
320 op=1
  .
  .

1000 sound DQ5/op,Y/8
1010 sound C5D,Y/8
1020 sound DQ5/op,Y/8
1030 sound C5D/op,Y/8
1040 sound DQ5/op,Y/4
1050 sound P,Y/4
1060 return
```

GLI STANDARD GRAFICI

Sono regole a cui bisogna uniformarsi per poter usare un unico complesso di programmi grafici su diversi elaboratori e con diverse periferiche.

AGM ASSOCIATION-ARCHIVIO EIDOS

Nel settore informatico un'esigenza pressante del mercato è sempre stata quella di poter utilizzare pacchetti software in grado di funzionare sul maggior numero di macchine possibile, anche di diverse marche. L'obiettivo è idealmente quello di rendere il software utilizzabile indipendentemente dall'hardware disponibile: questo è il concetto di standard.
La motivazione principale che ha portato alla definizione di uno standard in informatica grafica è stata la volontà di ridurre i costi in due fasi critiche: l'addestramento dei programmatori e il riutilizzo dei programmi su differenti sistemi; con una locuzione unica: portabilità dell'intero software. Definire uno standard non è però facile, soprattutto se si richiedono requisiti spesso contrastanti, quali la massima generalità e un buon livello di efficienza. Così a lato dell'accezione di standard quale modello stabilito d'autorità, si è imposta anche quella di standard quale esempio proposto da un costruttore leader nel settore che ha ottenuto un largo consenso, o standard *de facto*, che ha portato a interpretare lo standard come possibilità di sostituzione tra elementi conformi, cioè fisicamente compatibili.

Nell'ambito informatico uno standard può essere stabilito da autorità internazionali quali l'ISO (International Standards Organization) o locali quali l'americano ANSI (American National Standards Institute) e il tedesco DIN (Deutsches Institut für Normung). Essi possono sia formalizzare semplicemente degli standard de facto, sia promuovere gruppi di lavoro autonomi che redigano proposte proprie.

Gli standard de facto

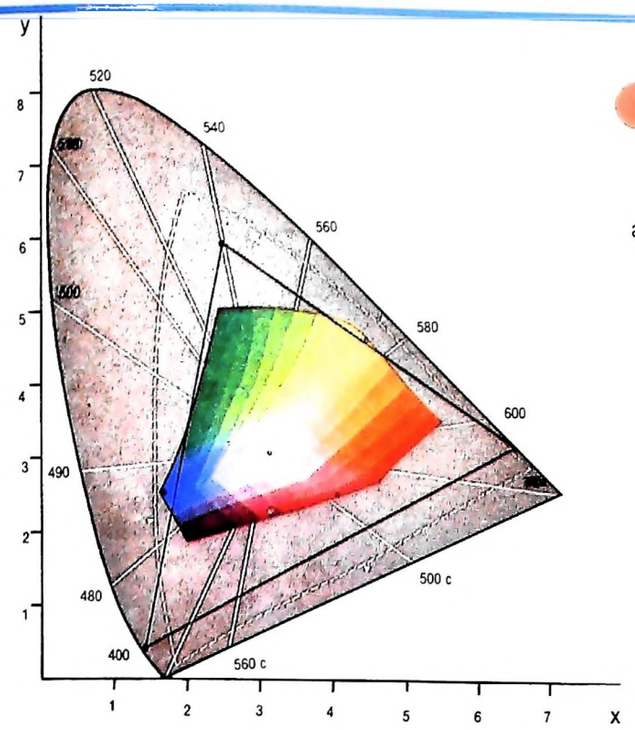
Vogliamo ora fornire qualche esempio di come si sono affermati alcuni standard de facto nel mondo della grafica computerizzata. A partire dalla fine degli anni '60 nel settore dei plotter la specifica "Calcomp compatibile", che indica un insieme di comandi grafici predisposti unicamente alla generazione di linee e testi su periferiche quali appunto i plotter, è diventata una garanzia per assicurare il corretto comportamento rispetto alle numerose interfacce hardware e pacchetti software disponibili sul mercato.
Di tutt'altra natura è stata la proposta della Tektronix che opera nel settore dei terminali video. L'idea fu quella di elaborare un software (Plot 10) che permettesse all'utente di manipolare con efficienza il tracciamento di linee sul video.
Questo pacchetto grafico venne successivamente integrato con funzioni di più alto livello, quali l'uso del colore, il calcolo di linee curve, la composizione di testi, la suddivisione in sottoparti dell'immagine.
Un ulteriore standard de facto risulta essere la libreria 2250/3250, un software grafico proposto dalla IBM. Esso è in grado di supportare vari tipi di input quali la tastiera, la penna ottica e la tavoletta grafica; peraltro essendo orientato ai soli terminali IBM resta escluso da una larga diffusione.
Queste proposte non sono buoni standard generali per diversi motivi. Innanzitutto esse sono vincolate alle caratteristiche hardware delle periferiche per le quali sono state pensate, per cui la casa che le ha elaborate non può correggerne i difetti presenti, pena la perdita dei propri vecchi clienti.
In secondo luogo, tali proposte sono generalmente nate o sono ancora con una visione bidimensionale e con scarse o nulle capacità di instaurare un rapporto interattivo con l'utente.
Infine le informazioni sia in ingresso che in uscita non raggiungono un sufficiente grado di indipendenza dalla periferica influendo pesantemente sulla portabilità dei programmi.

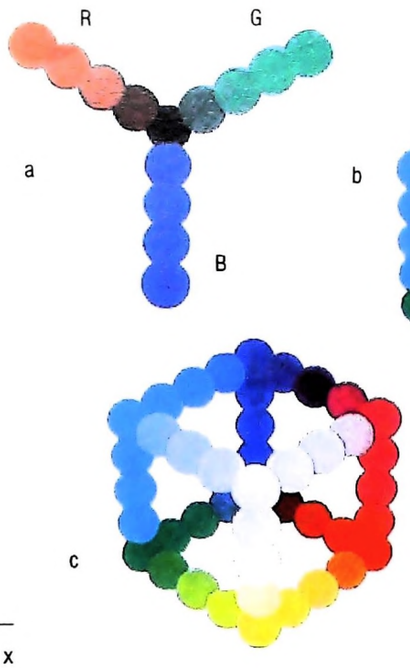
Le proposte di standardizzazione

Mentre i diversi costruttori cercavano di imporre il proprio standard o di adattarsi con un personale linguaggio di comandi grafici che fosse estensione di uno standard de facto, nelle commissioni ANSI e DIN venivano definendosi le due principali proposte generalizzate: rispettivamente il SIGGRAPH GSPC (Graphical Standards Planning Committee) CORE nel periodo '73-'79 e il GKS (Graphical Kernel System) dal '75 ad oggi.
Entrambi si basano su un insieme di concetti comuni, ma differiscono in numerosi punti ove sostanzialmente il GKS, definito successivamente, cerca di correggere alcune debolezze ed ambiguità riscontrabili nel GSPC CORE.
Gli aspetti comuni si possono riassumere con le seguenti funzionalità base: per generare linee, simboli, testi, poligoni;

Il computer consente di utilizzare stili grafici diversi. L'immagine della pagina precedente, per esempio, è stata ottenuta utilizzando uno stile pittorico; questa, invece, presenta uno stile orientato verso l'iperrealismo, tanto da dare la sensazione di una immagine fotografica.

ADM ASSOCIATION-ARCHIVIO EIDOS





A sinistra il diagramma della cromaticità secondo la definizione della CIE (Commission Internationale de l'Eclairage): i tre colori primari (rappresentati dagli assi X, Y e Z) possono essere combinati per definire tutte le sensazioni di colore che possiamo sperimentare. I tre disegni qui accanto presentano il modello RGB (Red, Green, Blue, ovvero rosso, verde e blu) per la specificazione dei colori in riferimento agli apparecchi di tipo televisivo a colori. Il modello usa un sistema di coordinate cartesiane: in a) si vedono i tre assi R, G e B; in b) i piani RB, RG e GR; in c) sono visualizzati gli spigoli del cubo RGB.

utilizzo di insiemi di attributi per ogni comando elementare in grado di modificarne la visualizzazione quali dimensioni, colori, orientamenti; suddivisione del disegno in sottofigure dette segmenti che possono essere rese visibili o invisibili a piacere e posizionate dove più interessa; trasformazione delle coordinate mondo, che sono quelle in cui l'utilizzatore intende esprimersi, in coordinate normalizzate, cioè tra 0 e 1, e quindi in coordinate della periferica tramite coppie di window-viewport, ciò che permette di usare nel programma valori indipendenti della periferica stessa; memorizzazione permanente su file di determinate parti della grafica generata. Dato che esiste una molteplicità di sistemi grafici aventi caratteristiche a volte molto diverse, i proponenti di entrambi gli standard hanno provveduto a fornire delle versioni parziali onde evitare la nascita e la proliferazione di dialetti di volta in volta comprendenti questa o quella caratteristica.
Gli aspetti peculiari che invece differenziano i due approcci sono: il GSPC CORE dispone di funzioni tridimensionali dotate di proiezioni parallele e prospettiche, e di estensioni per i terminali a scansione lineare come il televisore; il GKS presenta una più precisa suddivisione delle funzioni di input secondo raggruppamenti e l'inquadramento dell'architettura dei sistemi grafici è basata sul concetto di workstation. Questo concetto consiste nell'attribuzione alla stazione di lavoro di quella parte del processo di visualizzazione che trasforma le coordinate normalizzate in coordinate proprie della periferica e viceversa per i dati in ingresso, e nella possibilità di dare diverse rese alla stessa primitiva, a seconda delle capacità della stazione, in maniera del tutto automatica. La pulizia indotta da una visione così omogenea e l'essere in grado di sfruttare appieno le prestazioni locali sempre maggiori che le stazioni hardware vanno via via assumendo obbligano a considerare questa filosofia l'unica adatta per ambienti di calcolo distribuiti.

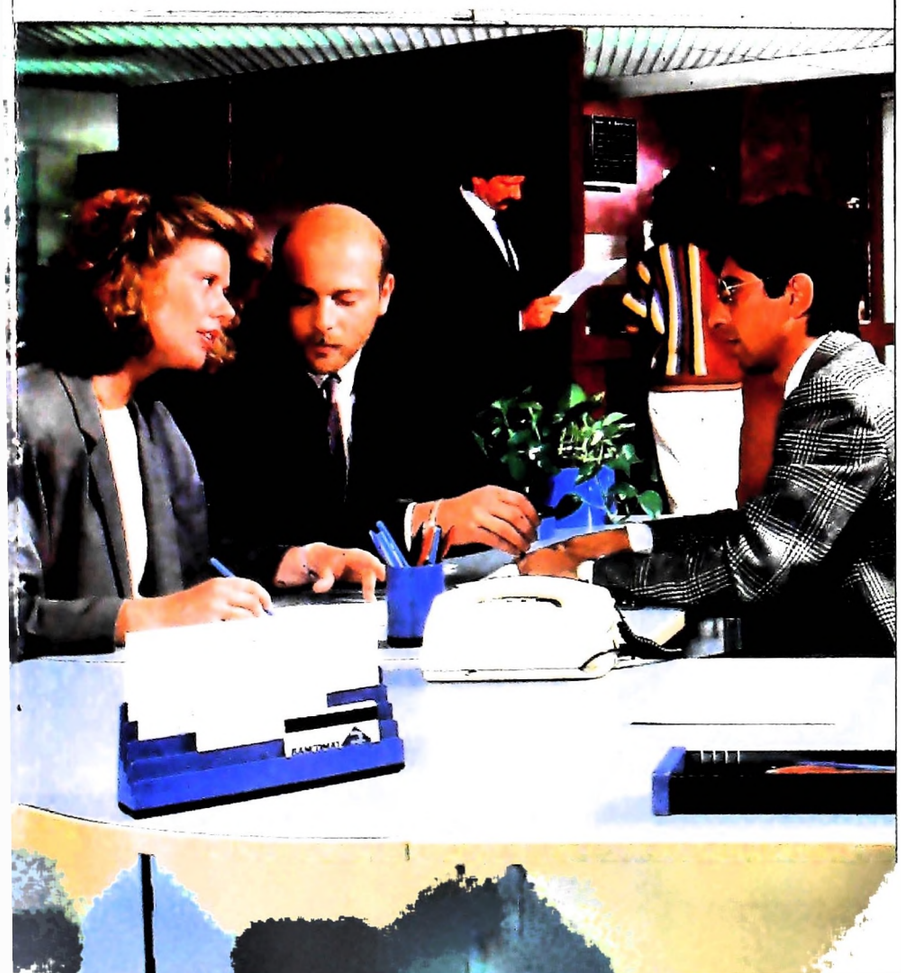
Standard grafici e linguaggi di programmazione

Un punto delicato nell'uso degli standard riguarda la loro interfaccia verso i vari linguaggi di programmazione, cioè come è possibile all'interno di un normale programma accedere alle funzioni grafiche componenti lo standard stesso.
Allo stato dell'arte, sono molto diversi i possibili approcci grafici tra i linguaggi più utilizzati quali PASCAL, FORTRAN, BASIC. Ciò si verifica sia per le caratteristiche intrinseche delle strutture dati che essi sono in grado di fornire, sia per la disponibilità degli stessi su classi di macchine molto diverse: BASIC sui personal computer, PASCAL e FORTRAN su medi e grandi calcolatori. Il FORTRAN e ancor più il PASCAL hanno a disposizione strumenti potenti che permettono una chiara "implementazione" delle primitive grafiche e la possibilità di definire ulteriori livelli di software utente. Invece, in BASIC, se si vuole cercare una minima aderenza ai concetti degli standard, si è costretti a simulare tutto ciò a scapito di una buona efficienza e leggibilità o si finisce con lo scrivere programmi di difficile comprensione.
Si verifica di conseguenza che le primitive solitamente disponibili sui personal sono decisamente meno sofisticate e molto differenziate da macchina a macchina.
Ciò creava diversi problemi quali ad esempio la necessità di duplicare e modificare tutti i comandi atti a realizzare un disegno se lo si desidera produrre contemporaneamente sullo schermo e su un plotter.
Al contrario, la visibilità diretta della memoria grafica che i personal permettono offre, se utilizzata da abili programmatori di base, di ottenere buone velocità di esecuzione e interessanti operazioni di manipolazione delle immagini già formate. Queste operazioni hanno sollevato varie perplessità nei proponenti degli standard e sono in corso anche tentativi per dare a questi ultimi un preciso inquadramento metodologico.

Sommario del Primo Volume

UN NUOVO MODO DI USARE LA BANCA.

Conto corrente più



TANTI PENSIERI IN MENO CON IL CONTO CORRENTE "PIU'" DEL BANCO DI ROMA.

Essere cliente del Banco di Roma vuol dire anche essere titolari del conto corrente "più". Un conto corrente più rapido: perché già nella maggior parte delle nostre filiali trovate gli operatori di sportello che vi evitano le doppie file.

Più comodo, perché potete delegare a noi tutti i vostri pagamenti ricorrenti: dai mutui all'affitto, dalle utenze alle imposte.

Più pratico, perché consente l'utilizzo del sistema di prelievo automatico Bancomat e l'ottenimento della carta di credito.

Più esclusivo, perché potete usufruire del servizio Voxintesi, attraverso il quale chiedere direttamente al nostro elaboratore il saldo del vostro conto corrente con una semplice telefonata: in qualsiasi ora come in qualsiasi giorno, anche festivo.

Più sicuro, perché con una minima spesa potrete assicurarvi contro furti e scippi mentre vi recate in banca o ne uscite.

Veniteci a trovare, ci conosceremo meglio.

BANCO DI ROMA
CONOSCIAMOCI MEGLIO.

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 informazioni rivolgersi a negozi contrassegnati da "Olivetti M10 Punto di Vendita" o inviare il coupon a: Olivetti Divisione Personal Computer, Via Meravigli 12, 20123 Milano

NOME/COGNOME

VIA/N.

CAP/CITTÀ

TELEFONO