13 CORSO PRATICO COL COMPUTER

421651

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

F4 F5 F6 F7

diretto da GIANNI DEGLI ANTONI

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

FABBRI EDITORI

A partire dal 25 giugno sarà in edicola il quinto numero di LIBRERIA DI SOFTWARE dedicato alla "Tabella elettronica"

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), VIRGINIO SALA, ADRIANO DE LUCA, 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

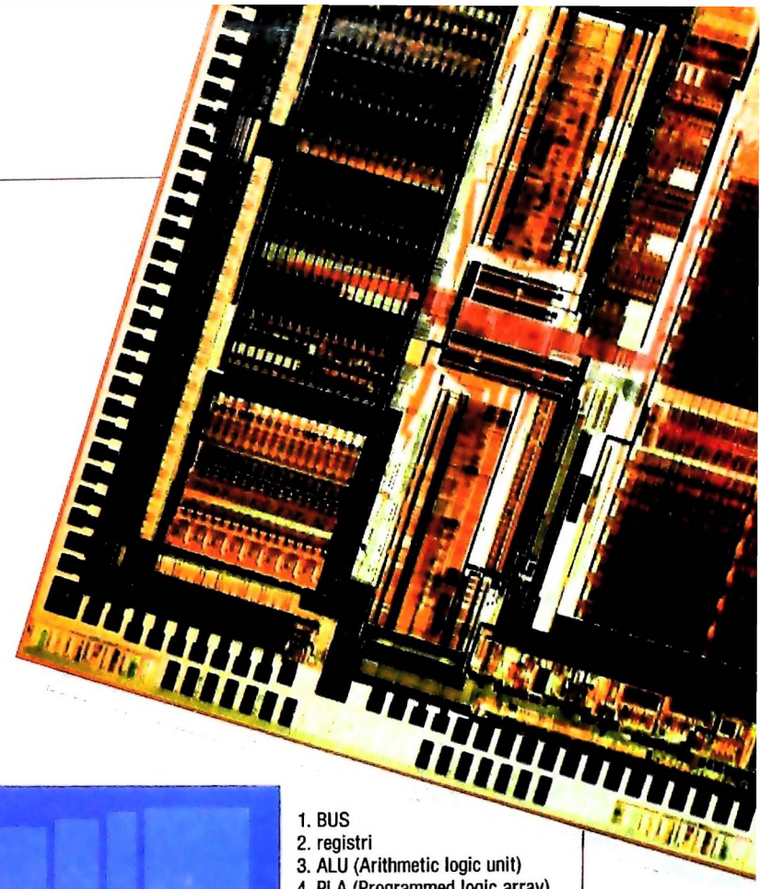
Impaginazione
BRUNO DE CHECCHI
PAOLA ROZZA

Programmazione Editoriale
ROSANNA ZERBARINI
GIOVANNA BREGGÈ

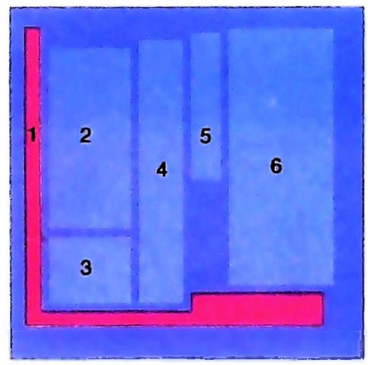
Segretarie di Redazione
RENATA FRIGOLI
LUCIA MONTANARI

IL BUS

Una struttura fondamentale nell'architettura dei microprocessori.



Un concetto fondamentale per i circuiti digitali moderni (e specialmente per i microprocessori) è quello di BUS. La sua definizione è la seguente:
gruppo di linee conduttrici che interconnettono diversi tipi di registri sia all'interno sia all'esterno dei microprocessori.
Esistono tre diversi tipi di BUS determinati in base al contenuto:
— BUS di dati
— BUS di indirizzi
— BUS di controllo.



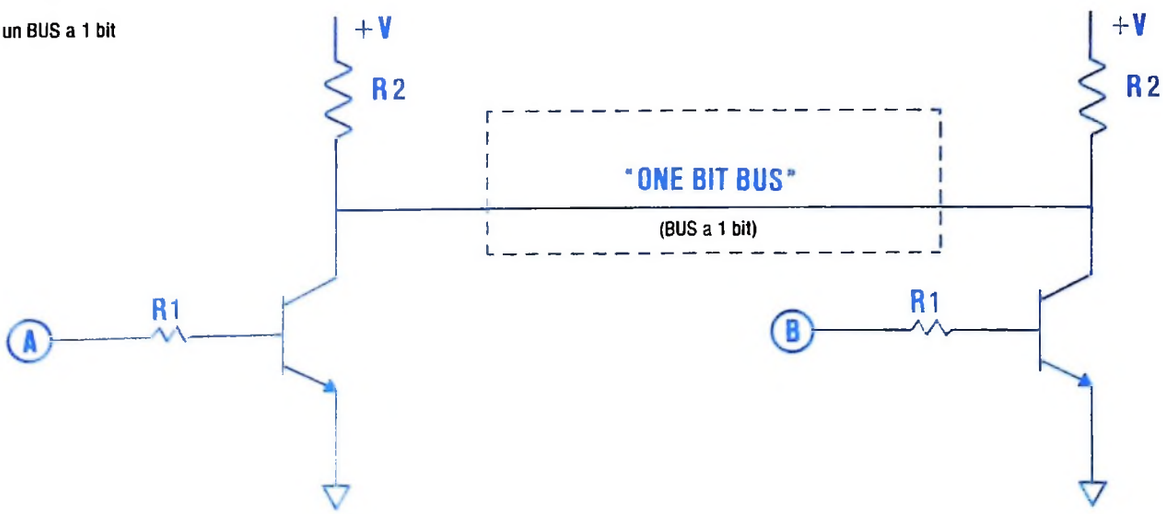
1. BUS
2. registri
3. ALU (Arithmetic logic unit)
4. PLA (Programmed logic array)
5. sequenziatore
6. microcodificatore della memoria di sola lettura

Architettura del microprocessore HP9000 a 32 bit della Hewlett-Packard. Esso comunica con il mondo esterno attraverso il suo BUS.

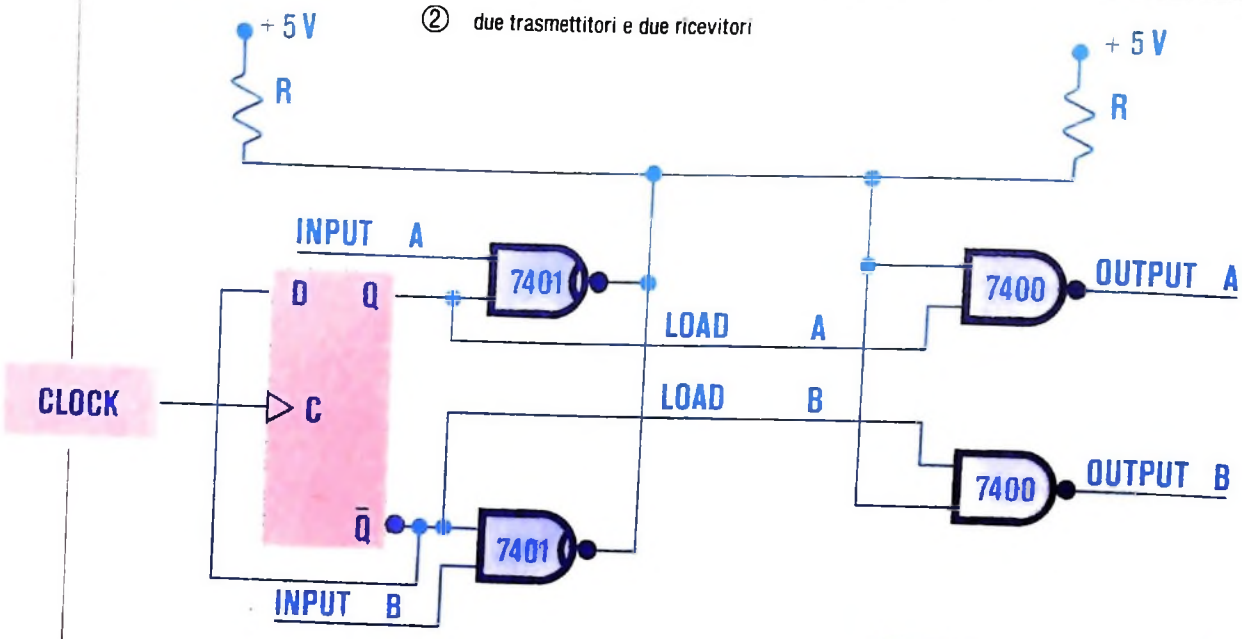
Tipi principali di BUS

Per quanto riguarda la loro realizzazione pratica, la distinzione non è però sempre così netta. Esistono infatti alcuni microprocessori moderni che usano solo un BUS di controllo più un altro BUS che riunisce due funzioni assieme, cioè trasporta dati e indirizzi alternativamente. Vi sono naturalmente diverse combinazioni possibili; ci limiteremo ad elencarne alcune, fra quelle più usate.
Ciò premesso, entriamo nel merito della descrizione del BUS cominciando a prendere in considerazione la figura 1. Essa mostra due transistor, montati secondo lo schema RTL (*Resistor Transistor Logic*), collegati allo stesso filo del BUS. Dalle caratteristiche dei transistor e dal modo in cui sono collegati al BUS discende che essi possono solo trasmettere e solo uno alla volta. Quindi, per completare la figura, aggiungeremo al termine del filo degli elementi ricevitori che, naturalmente, possono essere più d'uno. Esiste però una restrizione a questo tipo di circuiti, che consiste in questo: in qualsiasi gruppo di elementi trasmettitori e ricevitori collegati fra di loro attraverso lo stesso filo del BUS, solo uno degli elementi

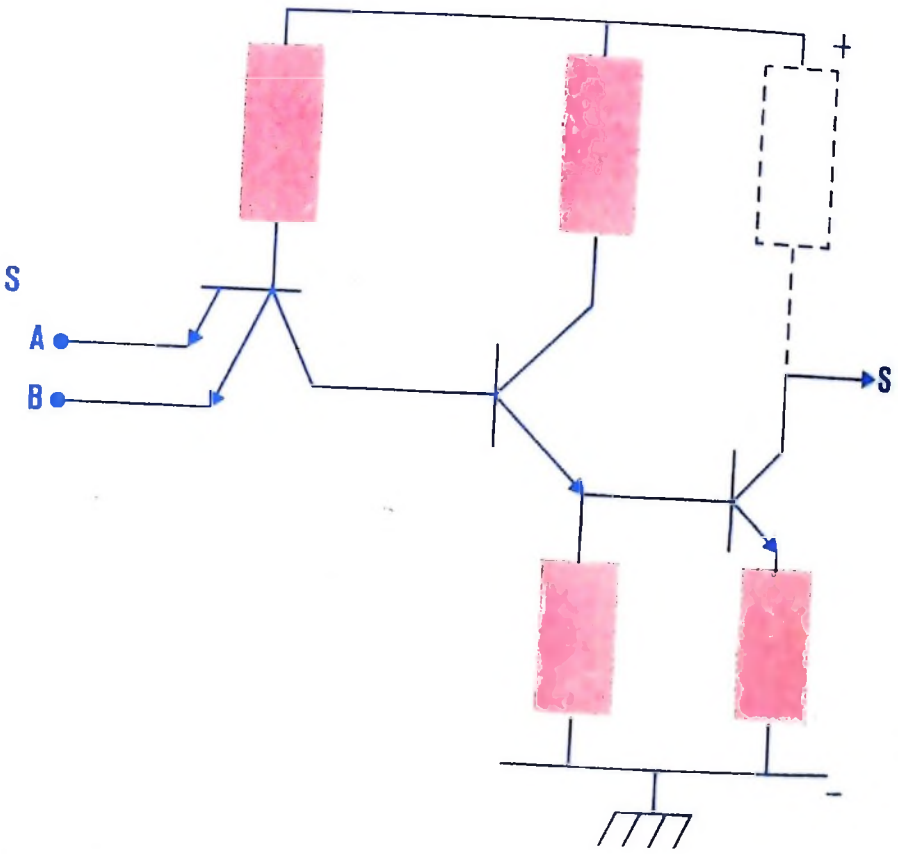
① un BUS a 1 bit



② due trasmettitori e due ricevitori



③ il 7401 visto internamente e la sua tabella di verità



A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

trasmettitori può essere attivo in un determinato istante, mentre uno o più elementi ricevitori lo possono essere contemporaneamente.

Vediamo ora l'esempio di due trasmettitori e due ricevitori temporizzati (figura 2) da un segnale detto di clock (orologio). Ricordiamo come sia buona norma, se non diversamente specificato, che l'informazione trasmessa non cambi stato al suo arrivo, anche se durante la trasmissione essa può essere opportunamente modificata per adattarla al mezzo di trasmissione.

Nella figura compaiono circuiti logici di tipo 7401, a collettore aperto (indicati con l'arco di cerchio all'uscita). Se analizziamo internamente tale tipo di circuito, con l'aiuto della tabella di verità (figura 3) vediamo che nel caso di entrata B = 1 l'entrata A cambia di stato, mentre con B = 0 ciò non si verifica; lo stesso processo avviene nel circuito NAND di uscita.

Continuando sempre nell'analisi della figura 2, il flip-flop che compare a sinistra è di tipo D, con l'uscita Q collegata all'entrata D, per cui quando nel clock abbiamo una transizione da 0 a 1 (ricordiamo che il segnale di clock è un susseguirsi di transizioni da 0 a 1 e da 1 a 0 con periodo fisso) il valore presente in D passa a Q e il suo valore negato, cioè l'uscita $\bar{Q}$ a sua volta, data la configurazione, si ripresenta in D. Segue che, alla prossima transizione del clock da 0 a 1, l'uscita Q cambia di stato e via di seguito per ogni transizione positiva (cioè da 0 a 1) del segnale di clock.

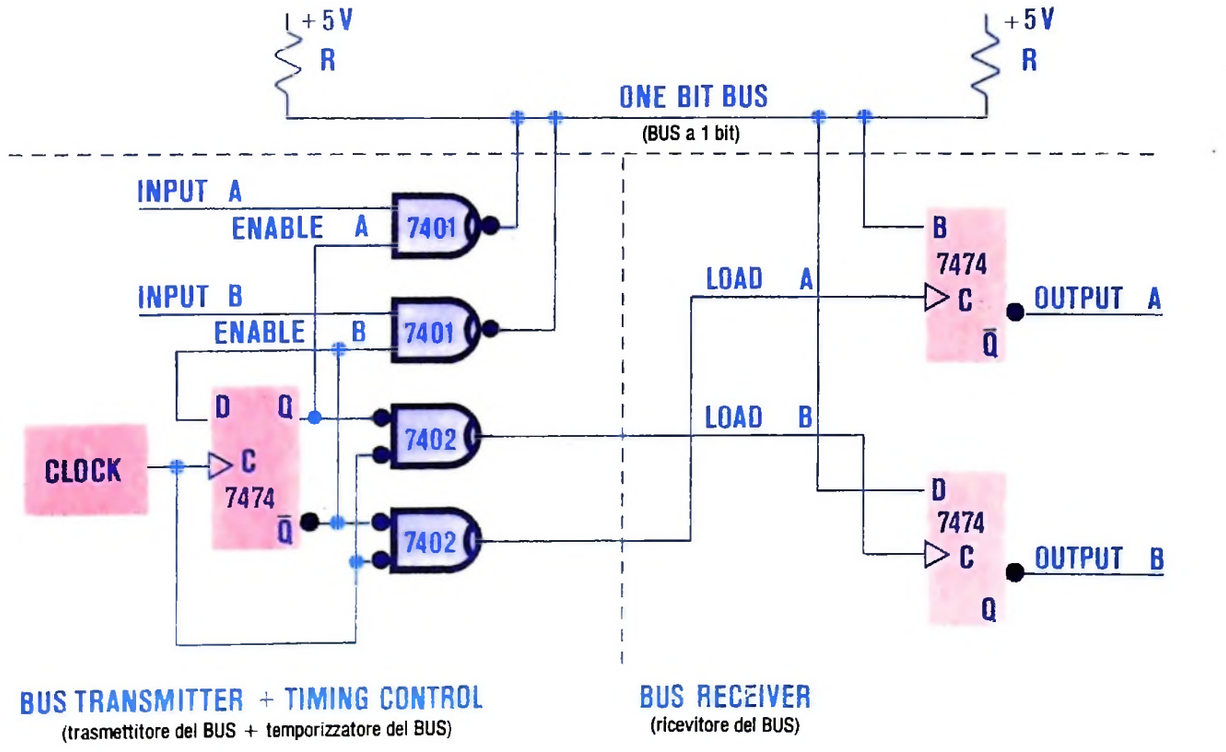
La trasmissione e la ricezione viene fatta in forma alternata

per ogni linea: quando il Q del flip-flop è uguale a 1 allora l'informazione che entra in INPUT A viene invertita dal circuito 7401, trasmessa sulla linea e ricevuta dal 7400; qui la fase viene reinvertita, per cui all'uscita A l'informazione viene ripresentata nella forma originale. Lo stesso succede nella linea B, quando l'uscita $\bar{Q}$ del flip-flop si trova nello stato 1.
È molto importante, per ciò che verrà spiegato nel prossimo futuro, notare i comandi LOAD A e LOAD B: queste sono due linee di controllo che permettono l'interconnessione del trasmettitore A col ricevitore A e del trasmettitore B col ricevitore B su un unico filo del BUS senza equivoco.

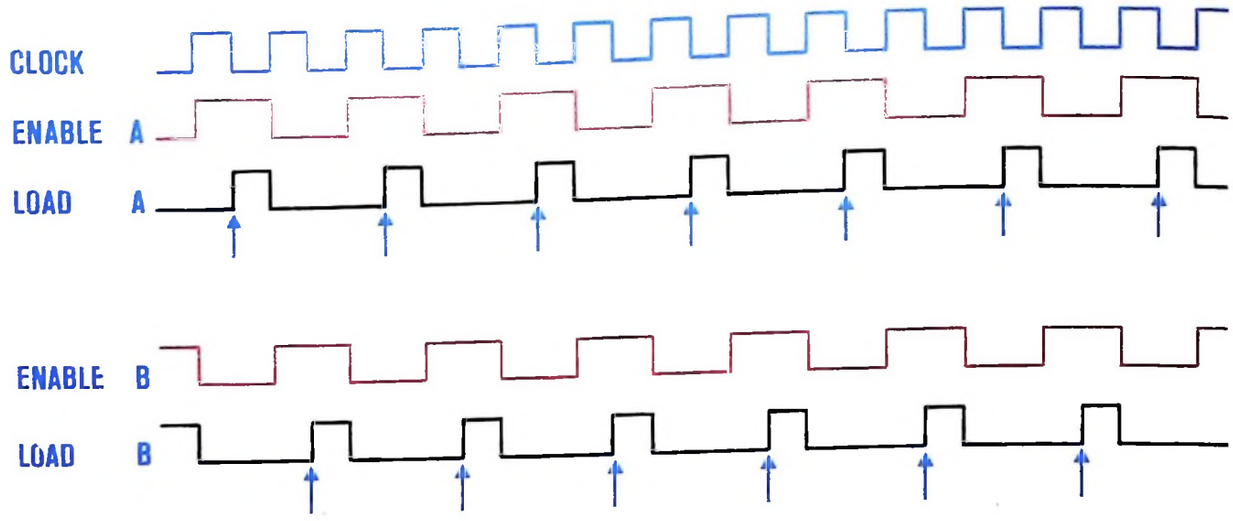
La logica del BUS

È interessante notare sin d'ora che, anche se ci troviamo di fronte a un circuito di trasmissione e ricezione in forma embrionale, il numero di linee necessarie rispecchia la logica del BUS: abbiamo una sola linea di dati ma due linee di controllo. Se aumentiamo il numero di elementi trasmettitori e ricevitori sulla stessa linea quest'ultima rimane invariata, mentre aumentano in proporzione le linee di controllo. D'altra parte, invece, se aumentiamo il numero di bit di dati, trasmessi contemporaneamente, aumenterà il numero delle linee dei dati, ma resteranno invariate le linee di controllo.
Per migliorare la nostra configurazione anteriore ed evitare in ogni modo la presenza di stati, anche se sbagliati, sugli elementi di uscita — i 7400 — passiamo alla configurazione mostrata nella figura 4. Cominciamo a suddividere gli elementi secondo la loro funzione: nella parte superiore della linea orizzontale tratteggiata c'è la linea del BUS, nella parte inferiore a sinistra della linea tratteggiata verticale ci sono i trasmettitori, sulla destra i ricevitori. Questi ultimi sono dei flip-flop di tipo D, che già conosciamo, ma in certi casi, secondo le circostanze, vengono chiamati LATCH per la funzione che hanno di ritenere un dato (*latch* vuol dire chiavistello).
In questo caso ognuno di essi ha il compito di immagazzinare l'informazione del suo trasmettitore e di non variarla durante il tempo in cui l'altra linea è in fase attiva. Vediamo ora come: prima di tutto, in questo caso, per poter rimettere in fase i dati trasmessi all'uscita li riprendiamo nel $\bar{Q}$ del flip-flop. In secondo luogo, abbiamo detto che il flip-flop D trasmette il dato presente sull'entrata D all'uscita Q quando c'è una transizione positiva nel clock: dopo questa operazione il dato in uscita non cambia anche se cambia l'entrata D fin quando non c'è un'altra transizione positiva nel clock. Apriamo una piccola parentesi per chiarire un problema tecnico: abbiamo detto finora che in un flip-flop D i dati in D passano in Q quando c'è una transizione sul clock. Ciò significa che, in seguito ai vari ritardi di trasmissione presenti nei circuiti, i dati devono essere presenti all'entrata D con un certo intervallo di anticipo, specificato dai fabbricanti di circuiti integrati, rispetto all'arrivo della transizione del clock.
Torniamo ora all'analisi della figura 4, e osserviamo l'andamento dei dati e dei comandi, tenendo presente il diagramma dei tempi della figura 5. Allo scopo di semplificare le cose supponiamo che alla prima transizione positiva del clock, da sinistra a destra nel diagramma, ci sia un 1 all'entrata D del flip-flop di controllo; nulla cambierebbe, come logica, se

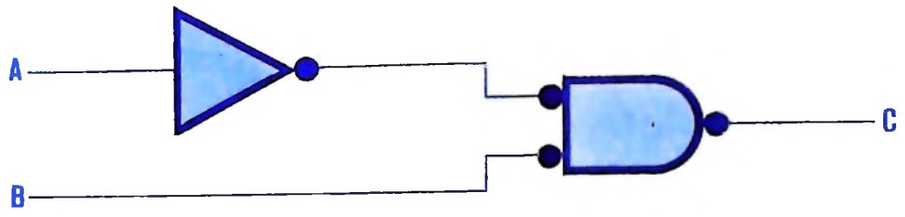
④ due trasmettitori e due LATCH per ricevitori



⑤ diagramma dei tempi del circuito di figura 4



⑥ circuito generatore di SPIT



⑦ rappresentazione grafica dello SPIT

vi fosse uno zero, all'uscita Q troviamo un 1 che pone a 1 la linea ENABLE A, cioè abilita il trasmettitore a trasmettere i dati e a presentarli all'entrata D del ricevitore.
All'uscita Q del controllo abbiamo uno 0 che, oltre a ripresentarsi all'entrata dello stesso controllo, mette uno zero anche all'entrata dell'AND 7402 inferiore. Dato che anche il clock è collegato alle due entrate dei 7402, quando esso diventa zero avremo nell'AND inferiore la coincidenza di due zeri in entrata, cioè un 1 in uscita, come si può vedere dalla freccina sulla linea LOAD A della figura 5. Questa transizione positiva nel clock del flip-flop A di ricezione permette il passaggio dei dati, già presenti all'entrata D, alla sua uscita OUTPUT A. Con la stessa sequenza vengono trasmessi i dati all'uscita OUTPUT B.

Conclusioni

Approfondiamo meglio l'analisi del circuito, per trarne alcune conseguenze interessanti:

a) Con la trasmissione positiva del clock si abilitano i circuiti di trasmissione a trasmettere i dati, mentre con la transizione negativa del clock si abilitano i ricevitori, presenti nel BUS, a immagazzinarli.
In questo modo si crea un tempo di ritardo fra trasmissione e ricezione dei dati, necessario per avere un corretto funzionamento.
b) Altro concetto importante è quello riguardante la necessità del clock per evitare errori di sincronismi.
Consideriamo il circuito della figura 6: in un primo momento abbiamo che A = 0 e B = 1, cioè C = 0 in uscita. Se in un determinato istante tutte e due le entrate cambiano contemporaneamente di stato, cioè A = 1 e B = 0, l'uscita dovrebbe continuare a essere C = 0, mentre a causa del ritardo prodotto dall'inversore all'uscita C si produce un impulso (figura 7) chiamato "SPIT", che può provocare errori nei circuiti che seguono.
Questo si può evitare usando un clock che dia un intervallo di tempo superiore al ritardo dato dall'inversore, per cui quando si verifica il valore di C lo si trova senz'altro stabile.

ALGEBRA DI BOOLE

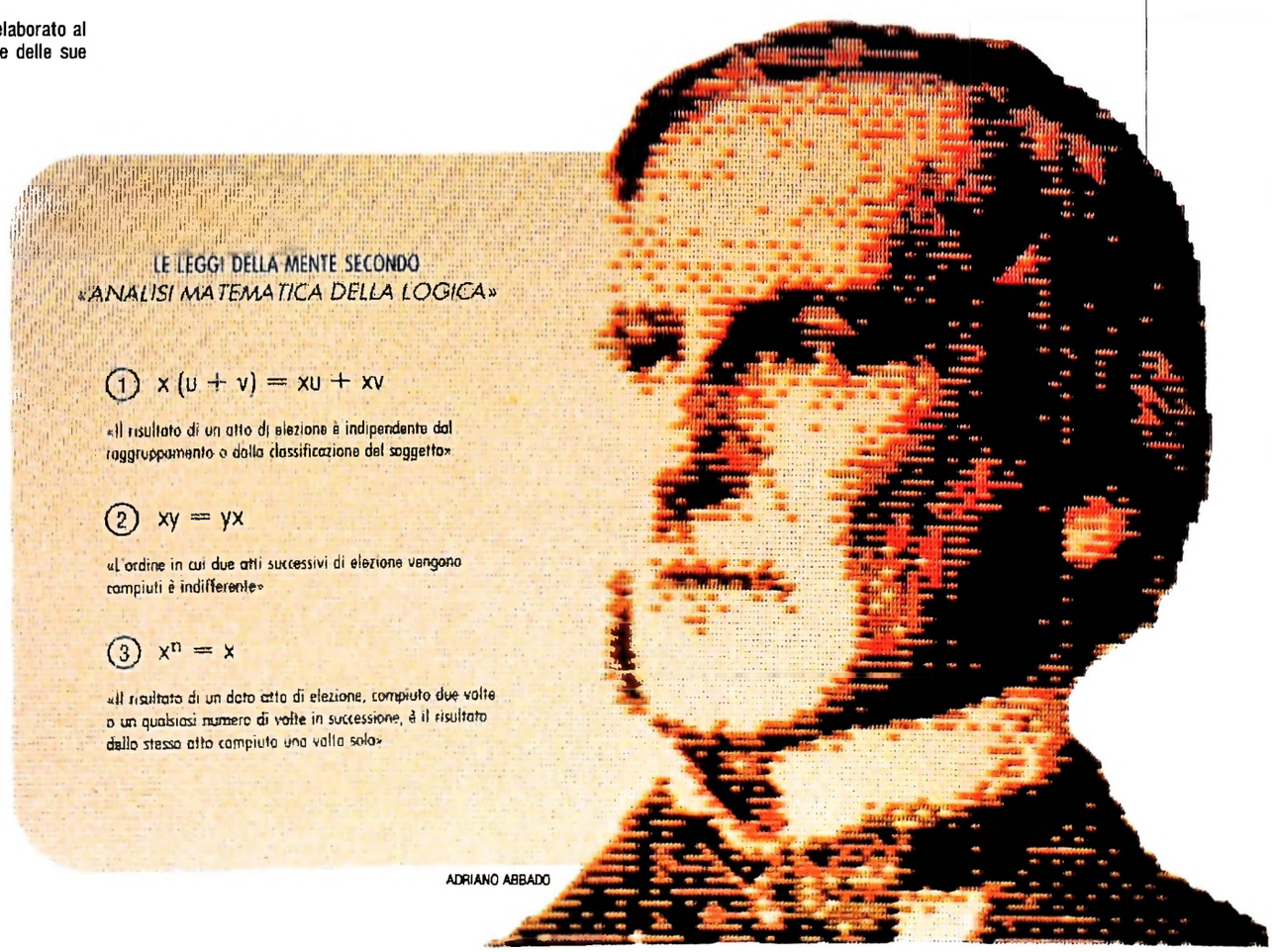
Una struttura algebrica di grande generalità, essenziale per capire il funzionamento dell'hardware di qualunque calcolatore.

Isoliamo una porzione del piano, che possiamo rappresentare come un quadrato: sarà il nostro universo di discorso. Consideriamo poi tutti i possibili insiemi di punti appartenenti a questo universo: li potremo rappresentare, come già sappiamo, mediante i diagrammi di Venn. Abbiamo già visto, parlando di insiemi, le operazioni fondamentali di unione, intersezione, complementazione. Notiamo ora che l'unione e l'intersezione di due insiemi entro il nostro universo di discorso danno ancora un insieme dell'universo; la complementazione di un insieme dell'universo dà ancora un insieme dell'universo. Diciamo che operazioni di questo tipo sono "operazioni interne" o "leggi di composizione interna".

Possiamo notare che queste operazioni godono di alcune proprietà interessanti: l'unione e l'intersezione sono commutative (come la somma e il prodotto fra numeri). L'unione di A e B è identica, in altre parole, all'unione di B e A: la verifica è immediata, con i diagrammi di Venn, e lo stesso si può dire dell'intersezione.

Ricordate la proprietà distributiva del prodotto rispetto alla somma, per i numeri? Dice che $a \times (b + c)$ è uguale ad $(a \times b) + (a \times c)$. Sostituite l'unione alla somma e l'intersezione al prodotto e avrete la formulazione della proprietà distributiva dell'intersezione rispetto all'unione fra insiemi: l'intersezione di A con l'unione di B e C è uguale all'unione dell'in-

Un ritratto di George Boole elaborato al calcolatore e la formulazione delle sue "leggi della mente".



ADRIANO ABBADO

L'algebra di Boole come struttura algebrica

(1) Per ogni x e y di B	$x \vee y = y \vee x$	PROPRIETÀ
(2) Per ogni x e y di B	$x \wedge y = y \wedge x$	COMMUTATIVA
(3) Per ogni x, y, z di B	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$	PROPRIETÀ
(4) Per ogni x, y, z di B	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$	DISTRIBUTIVA
(5) Per ogni x di B	$x \vee 0 = x$	
(6) Per ogni x di B	$x \wedge 1 = x$	
(7) Per ogni x di B	$x \vee x' = 1$	
(8) Per ogni x di B	$x \wedge x' = 0$	
(9)	$0 \neq 1$	

I simboli:
- $\vee$ si legge «unione»
- $\wedge$ si legge «intersezione»
- ' si legge «complemento di»
- 0 indica l'elemento nullo o zero
- 1 indica l'elemento unità o universo

Degli assiomi che caratterizzano l'algebra di Boole è possibile dare molte formulazioni equivalenti: qui vicino una delle più usate tra quelle moderne, mentre nella pagina a fronte una molto simile a quella originale di Boole, risalente alla fine del secolo scorso.

tersezione di A e B con l'intersezione di A e C. Anche qui il modo migliore per verificare questa affermazione è provare con i diagrammi di Venn qualche esempio.
Per le operazioni aritmetiche non vale il viceversa: la somma non gode di una proprietà distributiva rispetto al prodotto; per l'unione rispetto all'intersezione fra insiemi, invece, vale anche il viceversa. L'unione di A con l'intersezione di B e C è uguale all'intersezione dell'unione di A con B e dell'unione di A con C. Ancora una volta, verificatelo con i diagrammi di Venn.
L'operazione di complementazione ci dà, a partire dall'insieme A, un insieme CA che gode ovviamente di due semplici proprietà: la sua unione con A dà l'universo di discorso, mentre la sua intersezione con A è vuota. È sufficiente ripensare alla definizione di "complemento" per verificare la correttezza di queste affermazioni.
L'insieme vuoto e l'insieme totale (l'universo di discorso) hanno un comportamento particolare: l'unione dell'insieme vuoto con un qualsiasi insieme X dà sempre come risultato l'insieme X stesso; l'intersezione dell'insieme totale con un qualunque insieme X dà sempre come risultato l'insieme X stesso.
Si noti che fra unione e intersezione esiste un rapporto particolare: se in un qualunque enunciato valido si sostituisce la parola "unione" con "intersezione" e a "insieme totale" si sostituisce "insieme vuoto" si ottiene un enunciato parimenti valido: è questo il principio della dualità.

Una struttura algebrica

Dove ci ha portato questo esame delle proprietà della nostra famiglia di insiemi e delle operazioni definite su di essa? A evidenziare una struttura algebrica particolare, che prende il nome di *algebra di Boole* o *algebra booleana*. L'algebra definita sulla nostra famiglia di insiemi è solo un caso di algebra di Boole, quindi è bene sottolineare la definizione astratta della struttura.
Si ha un'algebra di Boole quando è dato un insieme non vuoto, contenente almeno due elementi distinti, su cui sono definite due leggi di composizione interna, indicate genericamente con i simboli + e · che godono della proprietà commutativa e della proprietà distributiva l'una rispetto all'altra; esistono inoltre gli *elementi neutri* rispetto alle due operazioni, indicati rispettivamente con 0 e 1 (tali cioè che $0 + x = x$ e $1 \cdot x = x$ per qualunque elemento x dell'insieme) e per ogni elemento x dell'insieme esiste il suo complemento x' (tale che $x + x' = 1$). Nel caso della famiglia di insiemi vista in precedenza, 0 e 1 sono l'insieme vuoto e l'insieme totale.
L'algebra di Boole è una *struttura* algebrica, una sorta di "teoria" astratta in cui non si fa riferimento alla natura degli elementi, né alla natura specifica delle operazioni definite su di essi, ma solo alle relazioni che intercorrono fra gli elementi e alle proprietà di quelle operazioni. Possono esistere, in altre parole, molte concretizzazioni diverse dell'algebra di Boole, ma per tutte varranno gli stessi teoremi dimostrabili in astratto a partire dai postulati che abbiamo elencato in precedenza. Così, per esempio, si è postulata l'esistenza di un elemento neutro per ciascuna operazione definita; si dimostra che, per ciascuna operazione, tale elemento neutro è unico. La dimostrazione, condotta nel caso astratto, varrà per qualunque esempio concreto di algebra di Boole.

Un'algebra di Boole a due elementi

Perché una struttura possa essere un'algebra di Boole bisogna (fra le altre cose) che l'insieme su cui è definita contenga *almeno* due elementi. Vediamo un caso molto semplice, in cui gli elementi sono, in effetti, *solo* due: 0 e 1 stessi.
Le operazioni (+ e ·) sono definite come segue: $0 + 0 = 0$; $0 + 1 = 1 + 0 = 1$; $1 + 1 = 1$; $0 \cdot 0 = 0$; $0 \cdot 1 = 1 \cdot 0 = 0$;

L'algebra astratta

Il termine "algebra" deriva da *Al-giabr al-Muqabala*, titolo di un'opera di Al-Kuwarizmi (dal cui nome deriva il termine "algoritmo"), matematico arabo, e stava originariamente a indicare il calcolo letterale e la risoluzione di equazioni algebriche (*al-giabr* significava il trasporto di un termine da un membro all'altro di una equazione). L'algebra moderna ha in apparenza ben poco a che vedere con l'algebra classica che si impara normalmente sui banchi della scuola media per arrivare, nel migliore dei casi, alla formula risolutiva dell'equazione di secondo grado. L'algebra moderna è "astratta" in quanto, a differenza della matematica classica, prescinde dalla natura degli oggetti che studia, e si concentra esclusivamente sulle loro relazioni e sulle proprietà delle operazioni che su di essi possono essere effettuate. Questo modo di "fare algebra" non esclude, ovviamente, il modo classico, ma rappresenta il passo successivo, uno stadio ulteriore di riflessione e di generalizzazione.

Si scopre che l'addizione sull'insieme degli interi relativi, il prodotto sui razionali non nulli, il prodotto sui reali non nulli, l'ordinaria somma fra polinomi in una indeterminata godono tutti delle stesse proprietà, cioè si comportano nello stesso modo: non è difficile pensare di trattarle come realizzazioni diverse di una medesima "legge di composizione" astratta, caratterizzata esclusivamente da quelle proprietà (quella a cui si arriva in questo modo, per esempio, è la nozione di "gruppo").

Il vantaggio di questo ulteriore passo sulla strada dell'astrazione è la generalità che si acquisisce: invece di tante trattazioni distinte, se ne sviluppa una sola, a un livello più alto, prescindendo dalla natura degli oggetti in gioco per concentrarsi sulle proprietà formali. Si evidenziano dei postulati, o assiomi, che identificano la "struttura" che si studia: tutto ciò che si deduce da questi assiomi risulterà applicabile a qualunque esemplificazione concreta di quella struttura, cioè a qualunque insieme di elementi che soddisfi agli assiomi. L'algebra booleana non è che una delle molte strutture studiate dall'algebra moderna.

Assiomi per l'algebra di Boole di A.N. Whitehead

(1) LEGGI GENERALI DELL'ADDIZIONE

$$a + b = b + a$$
$$a + b + c = (a + b) + c = a + (b + c)$$

(2) LEGGE SPECIALE DELL'ADDIZIONE

$$a + a = a$$

(3) DEFINIZIONE DELL'ELEMENTO NULLO

$$a + 0 = a$$

(4) LEGGI GENERALI DELLA MOLTIPLICAZIONE

$$c\,(a + b) = ca + cb$$
$$(a + b)\,c = ac + bc$$

(5) LEGGI SPECIALI DELLA MOLTIPLICAZIONE

$$ab = ba$$
$$abc = ab.c = a.bc$$
$$aa = a$$

(6) LEGGE DI «ASSORBIMENTO»

$a + ab = a$ [La legge (2) è un caso particolare di questa legge].

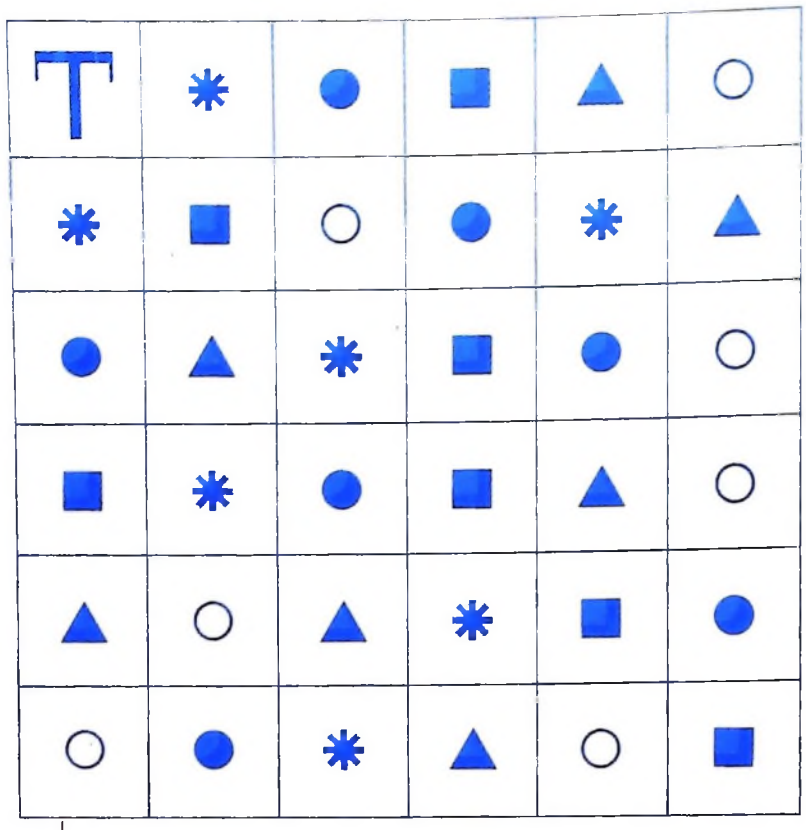
(7) DEFINIZIONE DELL'UNIVERSO

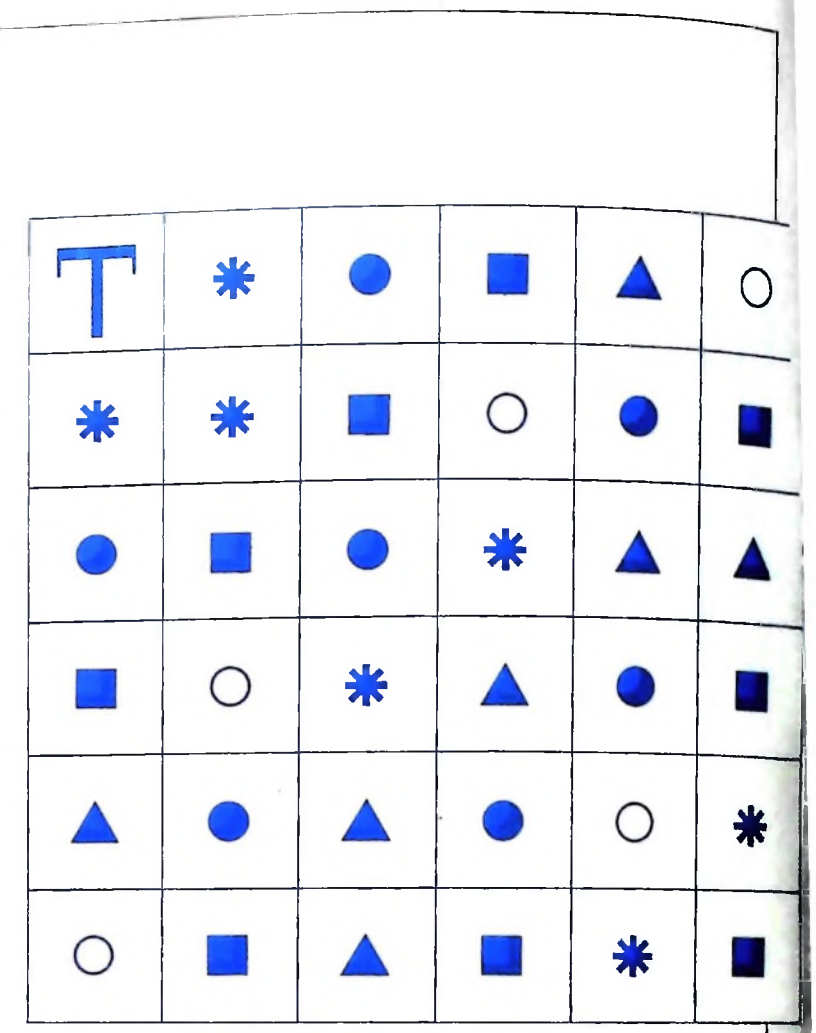
$ai = a$ [L'Universo è denotato da i].

(8) ELEMENTO SUPPLEMENTARE

L'elemento b è detto supplementare dell'elemento a se

$$a + b = i \text{ e } ab = 0$$





Rappresentazioni di due leggi di composizione (due corrispondenze che a una coppia di elementi ne associano un terzo). Queste due leggi in particolare sono descritte solo mediante tabelle che indicano, dato il primo e il secondo elemento della coppia, qual è il risultato.
Si può notare che nella prima il quadrato è elemento neutro, ma solo a sinistra, cioè quando è il primo fattore, mentre la seconda è commutativa ma non ha elemento neutro.

$1 \cdot 1 = 1$. (Non lasciatevi trarre in inganno: non sono esattamente identiche alla somma e al prodotto binari.) Il senso delle operazioni vi riuscirà chiaro se pensate a 1 come insieme totale e a 0 come insieme vuoto (l'intersezione dell'insieme vuoto con l'insieme totale è l'insieme vuoto, ..., l'unione dell'insieme totale con l'insieme totale è ancora l'insieme totale ecc.).
Le operazioni sono leggi di composizione interna: il risultato della loro applicazione a elementi dell'insieme dà ancora elementi dell'insieme. Inoltre 0 è l'elemento neutro rispetto alla "somma" ($0 + 0 = 0$; $0 + 1 = 1$) mentre 1 è l'elemento neutro rispetto al "prodotto" ($1 \cdot 0 = 0$; $1 \cdot 1 = 1$); e non è difficile verificare la validità della proprietà distributiva di ciascuna operazione rispetto all'altra. Infine 0 è il complemento di 1 (e viceversa). Il gioco è fatto: tutti i postulati dell'algebra di Boole sono soddisfatti.

Le algebre di Boole

L'algebra di Boole, struttura astratta, può dunque avere numerose "incarnazioni" in insiemi di elementi di natura anche molto diversa, e in ciascun caso le leggi di composizione interna saranno peculiari. Vedremo nel seguito due applicazioni dell'algebra di Boole strettamente pertinenti al tema dell'elaborazione delle informazioni: l'algebra delle proposizioni (una realizzazione dell'algebra booleana in campo logico) e l'algebra dei circuiti (che è in effetti alla base della progettazione dell'hardware dei computer. Il riquadro in questa pagina mostra un altro esempio curioso (per chi ama la matematica dilettevole).

Una curiosa algebra booleana

Tra le molte possibili realizzazioni della struttura astratta dell'algebra booleana, ne ricordiamo una presentata nel 1969 da Martin Gardner fra i suoi "Giochi matematici" (si veda "Le Scienze" n. 9, maggio 1969). Consideriamo l'insieme dei divisori del numero 30 (cioè dei numeri che dividono 30 esattamente, senza resto): {1, 2, 3, 5, 6, 10, 15, 30}. 1 e 30 sono divisori "impropri", ma rispondono perfettamente alla definizione. La "somma" dell'algebra booleana fra due di questi numeri sia il loro minimo comune multiplo; l'intersezione il massimo comun divisore. 30 è l'insieme totale, 1 è l'insieme "vuoto" (lo 0). Il complemento di un numero a è 30/a. Potete verificare da voi che le operazioni sono commutative e distributive l'una rispetto all'altra. Con lo stesso criterio, si può verificare che è un'algebra booleana l'insieme dei divisori di 210, cioè l'insieme {1, 2, 3, 5, 6, 7, 10, 14, 15, 21, 30, 35, 42, 70, 105, 210} sotto le stesse operazioni (l'insieme totale diventa 210, il complemento di a sarà 210/a).

Lezione 12

Completata questa dodicesima lezione del Corso di Programmazione e BASIC, siete in grado di eseguire gli esercizi
MINMXT.DO
MINMXP.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.

Cercheremo ora di scrivere un programma che ordina un certo numero di valori, forniti disordinati, in modo che alla fine risultino disposti per valore crescente.
Così, a partire da un insieme di valori, come per esempio:

3 1 7 1 1 4

produca la successione:

1 1 3 4 7 7.

Anzi, poiché i caratteri dell'alfabeto sono rappresentati all'interno del calcolatore come valori numerici, e tale rappresentazione usa valori crescenti secondo l'ordine alfabetico (la 'A' è minore della 'B', che a sua volta è minore della 'C', e così via), potremo porci il problema di ordinare lettere dell'alfabeto fornite in un ordine qualsiasi, esattamente nello stesso modo.
Se, per esempio, le lettere fornite sono le seguenti:

c r e z a d a

il programma le dovrà ordinare nel modo seguente:

a a c d e r z

Le lettere da ordinare verranno fornite dall'utente e quindi inserite in un array di stringhe.
Esistono molti algoritmi per risolvere questo problema. Uno dei più semplici è il cosiddetto algoritmo di ordinamento (in inglese sort) a interscambi:

- partendo dal primo elemento dell'array, esamina ogni successiva coppia di elementi;
- se la coppia è già ordinata passa ad esaminare la successiva; in caso contrario scambia i due elementi e ricorda che uno scambio è stato effettuato;
- vengono ripetute più passate dell'array fino a quando ne viene effettuata una senza scambi di elementi; ciò consente infatti di garantire che tutti gli elementi sono effettivamente in ordine.

Se le lettere da ordinare sono quelle indicate in precedenza, la prima passata produrrà in successione le seguenti configurazioni (in maiuscolo è mostrata la coppia esaminata):

```
C R e z a d a  (nessuno scambio)
c E R z a d a  (scambio)
c e R Z a d a  (nessuno scambio)
c e r A Z d a  (scambio)
c e r a D Z a  (scambio)
c e r a d A Z  (scambio)
```

Poiché sono stati effettuati scambi dovremo ripetere una successiva passata, che produrrà l'ordine seguente:

c e a d a r z

Altre successive passate verranno effettuate fino a produrre l'ordinamento definitivo. Il numero di passate sull'array non è noto a priori e dipende, oltre che dal nu-

mero di elementi da ordinare, anche da quanto sono disordinati. Esaminiamo dunque come si presenta l'algoritmo scritto in Pascal:

- REPEAT
  - Poni l'indicatore di scambio avvenuto uguale a "no"
  - FOR I:=1 TO N—1
    - IF elemento di indice I › elemento di indice I+1 THEN
      BEGIN
        - Scambia i due elementi
        - Poni l'indicatore di scambio avvenuto a "sì"

      END

UNTIL l'indicatore di scambio avvenuto è uguale a "no"

Si noti in particolare che nell'istruzione FOR l'indice varia da 1 al numero di elementi da ordinare meno 1. L'indice infatti è posizionato sul primo elemento della coppia che si confronta.
Come si vede meglio nella successiva istruzione IF, il confronto avviene tra l'elemento di indice I e l'elemento di indice I+1.
Se pertanto il valore finale dell'indice fosse il numero degli elementi da ordinare, si arriverebbe a confrontare l'ultimo elemento dell'array con uno successivo che evidentemente risulta inesistente, causando così un errore di tipo BS (violazione di "confini").
Si noti inoltre che valori uguali, se presenti, non vengono scambiati.
Vediamo quindi come si realizza il programma in BASIC:

```
1 CLS
5 PRINT "      PROGRAMMA DI ORDINAMENTO"
7 PRINT
10 INPUT "Numero elementi da ordinare";N
20 DIM L$(N)
30 FOR I=1 TO N
40 INPUT "Lettera";L$(I)
50 NEXT I
60 REM Inizio ordinamento
70 LET S$="no"
80 FOR I=1 TO N-1
90 IF L$(I)<=L$(I+1) THEN 150
100 REM Istruzioni di scambio
140 LET S$="si"
150 NEXT I
160 IF S$="si" THEN 70
170 FOR I=1 TO N
180 PRINT L$(I)
190 NEXT I
```

Come al solito, seguendo la filosofia dello sviluppo TOP DOWN abbiamo scritto lo schema generale del programma, indicando con un commento le parti più di dettaglio, che verranno successivamente costruite.
Vediamo dunque in che modo realizzare lo scambio; sarà necessario usare una variabile d'appoggio per ricordare la lettera contenuta in uno dei due elementi dell'array nel modo seguente:

- Assegna l'elemento di indice I alla variabile d'appoggio
- Assegna l'elemento di indice I+1 all'elemento di indice I
- Assegna la variabile d'appoggio all'elemento di indice I+1

Possiamo quindi costruire le istruzioni mancanti:

```
100 REM Istruzioni di scambio
110 LET T$=L$(I)
120 LET L$(I)=L$(I+1)
130 LET L$(I+1)=T$
```

Come memorizzare dati nei programmi

Supponiamo ora di non voler ordinare dati ogni volta differenti, ma al contrario di avere l'esigenza di ordinare la stessa sequenza di dati. Potrebbe accadere per esempio per un programma dimostrativo da mostrare a un potenziale acquirente: poiché è la funzionalità di ordinamento che ci interessa, non vogliamo ripetere a ogni esecuzione l'inserimento dei dati e ci farebbe invece comodo che il programma ne usasse un gruppo definito una volta per tutte. Si tratta in sostanza di definire un insieme di dati invariabili, che il programma ricorda da un'esecuzione all'altra: parleremo per questo di costanti.
In BASIC si realizzano nel modo seguente:

```
1 CLS
5 PRINT "      PROGRAMMA DI ORDINAMENTO"
7 PRINT
10 READ N
20 DIM L$(N)
30 FOR I=1 TO N
40 READ L$(I)
50 NEXT I
60 REM Inizio ordinamento
70 LET S$="no"
80 FOR I=1 TO N-1
90 IF L$(I)<=L$(I+1) THEN 150
100 REM Istruzioni di scambio
110 LET T$=L$(I)
120 LET L$(I)=L$(I+1)
130 LET L$(I+1)=T$
140 LET S$="si"
150 NEXT I
160 IF S$="si" THEN 70
170 FOR I=1 TO N
180 PRINT L$(I)
190 NEXT I
200 DATA 7,"t","s","b","h","a","e","r"
```

L'istruzione DATA ha l'effetto di memorizzare i valori costanti indicati in un'area di memoria che sarà poi "letta" grazie all'istruzione READ.
Quest'ultima assegna alle variabili specificate i valori costanti memorizzati tramite l'istruzione DATA. Tali valori vengono letti nell'ordine in cui sono stati memorizza-

ti e vengono ugualmente assegnati alle variabili nell'ordine indicato.
Naturalmente bisogna prestare attenzione alla congruenza tra le costanti assegnate e le relative variabili: costanti numeriche devono essere assegnate a variabili numeriche, mentre costanti stringhe devono essere assegnate a variabili stringhe, come nell'esempio seguente:

```
10 READ N
15 DIM C$(N),P(N)
20 FOR I=1 TO N
30 READ C$(I),P(I)
40 NEXT I
50 DATA 3,"caffe'",500,"the",700,"gin",1500
```

In uno stesso programma possono comparire più istruzioni DATA e più istruzioni READ. Le istruzioni DATA hanno comunque l'effetto di memorizzare dati in sequenza, qualunque sia il loro numero e la loro posizione nel programma. Pertanto i due "frammenti" di programma seguenti si equivalgono:

```
10 READ A,B$,C$,P
20 PRINT A;B$;C$;P
30 DATA 10,"AAAA","bbb",34
```

```
10 READ A
20 DATA 10
30 DATA "AAAA"
40 DATA "bbb",34
50 READ B$,C$
60 READ P
70 PRINT A;B$;C$;P
```

Cosa abbiamo imparato

In questa lezione abbiamo visto:

- Il concetto di ordinamento di un array
- L'algoritmo di ordinamento a interscambi
- La realizzazione BASIC dell'algoritmo di ordinamento per interscambi
- Il concetto di costante
- La realizzazione BASIC di costanti tramite le istruzioni DATA e READ
- Il concetto di rappresentazione interna di valori
- Le funzioni ASC e CHR$

LA STRATEGIA TUTORIALE

La prima e più sperimentata metodologia di applicazione dell'informatica alla didattica.

Dopo aver presentato gli aspetti tecnologici, che vengono anche indicati con il nome inglese di "hardware", prendiamo ora in considerazione il problema delle metodologie applicate nell'insegnamento tramite l'informatica.
Si tratta di un campo molto vasto: molte strade sono state battute, con risultati diversi, ma anche se il terreno è ancora in gran parte da esplorare qualcosa è stato concretizzato.
Vediamo di fare un bilancio, sia pure provvisorio.

Panoramica sui diversi metodi

Il metodo più noto, forse il primo ad essere impiegato, è il metodo *tutoriale*, nel quale vengono presentate all'allievo porzioni di materia, intervallate da controlli dell'apprendimento, in base al risultato dei quali vengono presentate le porzioni successive.
Un altro metodo molto usato, e abbastanza semplice, è quello delle *esercitazioni* (in inglese "drill and practice"), dove all'allievo vengono proposti esercizi da svolgere, graduati secondo difficoltà crescenti.
Con la *simulazione* l'elaboratore diventa una macchina per eseguire esperimenti nei settori più diversi ad esempio la fisica, la chimica, le scienze naturali, l'economia ecc.
È possibile così compiere sul video esperimenti che nella realtà risulterebbero lunghi, difficili, costosi o assolutamente irrealizzabili.
Affronteremo l'esame di questi tre metodi più oltre.
Vediamo prima gli altri metodi, con una presentazione più generale, ma abbastanza precisa.
Anzitutto l'*indagine* (in inglese "inquiry"): è una metodologia veramente innovativa e tuttora in fase di sperimentazione. L'allievo ha la possibilità di interagire con l'elaboratore per organizzare l'insieme di informazioni necessarie al raggiungimento di un obiettivo che lui stesso ha fissato. L'obiettivo tuttavia può anche essere assegnato dal sistema attraverso problemi da risolvere, mediante metodi e strumenti messi a disposizione dal sistema stesso. È prevista l'utilizzazione anche di strumenti diversi dall'elaboratore (diapositive, film, videodisco ecc.), ma gestiti attraverso l'elaboratore stesso.
Lo scopo di questa metodologia è quello di sviluppare capacità critiche, primo passo verso un apprendimento guidato dall'allievo. La realizzazione impiega anche metodi già applicati in ricerche di Intelligenza Artificiale, in genere però molto costosi dal punto di vista dell'impiego di risorse (ad esempio la memoria).
Si tratta comunque di una metodologia molto aperta a sviluppi futuri: attualmente sono in corso parecchie ricerche e sperimentazioni in questa direzione.
C'è poi il metodo della *risoluzione di problemi* (in inglese "problem solving") che aiuta l'allievo, come dice il nome, a risolvere problemi diversi e ad ampliare le proprie conoscenze. Attraverso questo metodo viene richiesta all'allievo la costruzione di un algoritmo per determinare la soluzione di un problema e che sia riconoscibile come tale dal sistema.
È chiaro che questa strategia lascia all'allievo grande libertà di scelta dei metodi risolutivi, purché tutti entro l'ambito di leggibilità da parte dell'elaboratore. L'allievo può fare uso di programmi esterni al sistema, per verificare le proprie ipotesi, o eseguire calcoli diversi in relazione ai problemi che deve risolvere. Il metodo lascia parecchio spazio alla creatività dell'allievo.
Per concludere citiamo il metodo dei *giochi* di abilità. L'allievo può essere posto in due condizioni diverse: avversario dell'elaboratore oppure suo insegnante istruttore. Nel primo caso si attua una sfida tra elaboratore e studente, con il rischio che lo studente perda sempre, se l'elaboratore è stato programmato a giocare usando una strategia sicuramente vincente.
Nel secondo caso l'elaboratore ha il ruolo di un allievo ignorante che deve imparare a giocare. In questo caso l'elaboratore deve "imparare" dallo svolgimento del gioco ad evitare di prendere decisioni errate.
L'esperienza è utile principalmente per l'analisi degli errori che l'allievo può fare, controllando quelli dell'elaboratore che sta imparando. Lo scopo è quello di guidare l'allievo a prendere delle decisioni corrette in vista del raggiungimento di un certo obiettivo.

La strategia tutoriale

Guardiamo ora più da vicino e più dettagliatamente il metodo *tutoriale* dall'inglese "tutor".
Questo metodo è impiegato ormai da lungo tempo nell'insegnamento mediante elaboratore, ed è tuttora uno dei più usati ed oggetto di ricerche e di sperimentazioni, poiché offre una varietà notevole di soluzioni.
Lo schema più semplice è quello di suddividere la materia in porzioni elementari e di presentarle poi allo studente, intervallate da domande di verifica dell'apprendimento.
Le domande hanno in genere risposte a scelta multipla, cioè

lo studente deve fare la sua scelta entro un gruppo di risposte presentate dall'elaboratore.
Se la risposta data è corretta viene presentata una nuova porzione di materia; se invece è sbagliata l'elaboratore interrompe la presentazione, e solitamente viene attivata una sequenza di ricupero e correzione degli errori.

Vantaggi e svantaggi

Il principale vantaggio di questo metodo è che ogni studente procede al ritmo che gli è proprio e riceve informazioni di ricupero degli errori abbastanza personalizzate.
Un altro grande vantaggio è che dal punto di vista psicologico, lo studente non viene giudicato da un professore e sembra perciò meglio disposto a correggere i propri errori. Su questo punto e sui vantaggi più generali dell'impiego dell'elaboratore, torneremo più avanti considerando il problema più generale della valutazione.
Gli svantaggi sono essenzialmente legati al modo di impostaEcco un esempio di interazione tra un elaboratore e uno studente nel corso di un programma di tipo tutoriale riguardante una esercitazione sui numeri primi. Dopo aver richiamata la porzione elementare di teoria, che qui è appunto la definizione di numero primo, l'elaboratore propone allo studente uno o più esercizi di verifica, dopodiché passerà a un altro aspetto di teoria.

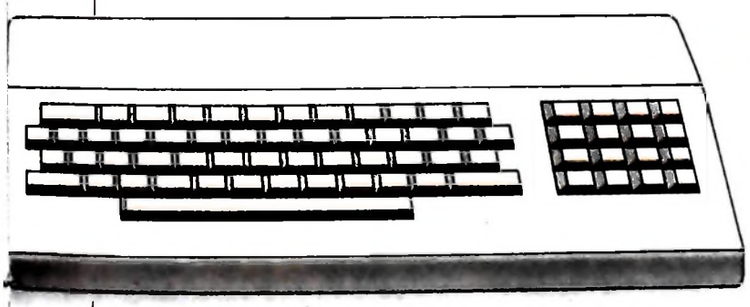
Come si vede, le domande dell'elaboratore sono a scelta multipla: lo studente deve indicare uno dei numeri proposti dal calcolatore. Questa procedura ha il pregio di essere molto semplice, ma a lungo andare, evidentemente, risulta noiosa e poco stimolante. Da notare, infine, che dopo la risposta esatta l'elaboratore riprende, per rinforzarli, i concetti introdotti nella porzione di teoria.

re il corso: una lezione potrebbe diventare monotona, fredda e quindi noiosa e demotivante per lo studente.
Uno dei problemi tuttora aperti è il tipo di risposta che deve dare lo studente.
È chiaro che, nel caso più semplice, la risposta *chiusa* (cioè a scelta multipla) è facile da realizzare, ma è piuttosto schematica e vincola il docente che prepara il programma entro risposte prestabilite, con il pericolo che la verifica risulti nel complesso piuttosto povera.

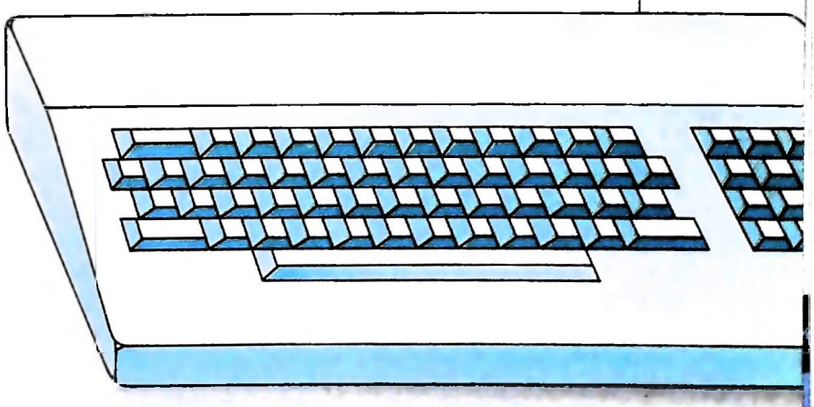
Inoltre lo studente potrebbe anche tirare a indovinare. Meglio sarebbe (e sono stati fatti esperimenti in questo senso) la risposta *aperta*, cioè impostata autonomamente dallo studente. L'elaboratore deve controllare la verità della risposta che, evidentemente, non può essere unica: solitamente viene controllata la presenza o meno di parole chiave.
Ma ciò potrebbe fare accettare come risposta valida anche una che valida non è, per esempio una che contenga le parole chiave, ma in un contesto di negazione.

PROVA ANCORA A RISPONDERE

37



BENE! ANDIAMO AVANTI...



Glossario

Accesso, tempo di - *il tempo necessario per la lettura o la scrittura di informazioni in un sistema di memoria. Nelle memorie principali (interne alla macchina) i tempi di accesso sono molto brevi; sono più lunghi per i sistemi di memoria secondaria, esterni alla macchina, come dischi o nastri magnetici.*

Accumulatore - *un registro nell'unità aritmetica e logica (ALU), usato soprattutto per le operazioni aritmetiche e per istruzioni attinenti a attività di input/output.*

ALGOL - *acronimo per ALGOrithmic Language, linguaggio algoritmico. Un linguaggio di programmazione di alto livello, progettato per applicazioni matematiche e, più in generale, scientifiche.*

COBOL - *acronimo di COmmon Business Oriented Language, indica un linguaggio di programmazione di alto livello, orientato specificamente alla programmazione per le attività commerciali.*

EFT - *acronimo di Electronic Funds Transfer, vedi Trasferimento elettronico di fondi.*

EPROM - *acronimo di Erasable Programmable Read Only Memory, memoria a sola lettura programmabile e cancellabile. È un tipo di ROM i cui contenuti possono essere cancellati e riprogrammati, mediante opportune apparecchiature (per la cancellazione si usa luce ultravioletta).*

Monitor - *1. un visualizzatore a tubo a raggi catodici, sostanzialmente simile a un televisore ma privo dell'apparato di ricezione dei programmi televisivi e specificamente studiato per la visualizzazione dell'uscita di un computer. Un monitor può essere monocromatico (caratteri bianchi, gialli o verdi su fondo nero) o a colori. - 2. software o firmware che funge da controllore, supervisore e verificatore dell'attività generale di un sistema.*

MOS - *acronimo di Metal-Oxide-Semiconductor, metallo-ossido-semiconduttore. Si riferisce alla configurazione a tre strati usata nella fabbricazione della struttura dei transistor a effetto di campo. La tecnologia MOS presenta bassissima dissipazione di potenza e rende possibile la realizzazione di circuiti ad alta densità di componenti senza problemi di surriscaldamento.*

MS-DOS - *sistema operativo sviluppato dalla società americana Microsoft per microcomputer a 16 bit, in particolare per l'IBM Personal Computer e per i suoi derivati.*

Nano - *prefisso che, nel sistema metrico decimale, indica un miliardesimo: un nanosecondo è pari a un miliardesimo di secondo.*

OCR - *acronimo di Optical Character Recognition, vedi Riconoscimento ottico di caratteri.*

Riconoscimento ottico di caratteri - *lettura da parte di una macchina di caratteri dattiloscritti o manoscritti. Esistono periferiche per computer in grado di riconoscere otticamente (con eccellenti prestazioni) testi dattiloscritti o manoscritti con calligrafia molto precisa e regolare: sono tuttavia ancora dispositivi complessi, costosi e poco duttili. Il riconoscimento ottico della normale calligrafia "corsiva" è ancora un traguardo lontano.*

Trasferimento elettronico di fondi - *trasferimento di somme di denaro da un conto bancario a un altro mediante tecnologie elettroniche. Per esempio, un terminale di computer in un negozio o in un grande magazzino può leggere la carta di credito di un cliente e, collegandosi al computer della banca del cliente e della banca del negozio, ordinare direttamente il trasferimento della somma necessaria per un acquisto, dal conto bancario del cliente al conto del negozio.*

User friendly - *espressione gergale che significa letteralmente "amichevole per l'utente". Viene usata genericamente per indicare una apparecchiatura, un programma, o qualunque aspetto di un sistema di calcolo, che risulti facile da apprendere e da maneggiare.*

Facciamo un esempio concreto: supponiamo che nella risposta debba comparire la parola chiave "energia" e lo studente risponda "non si tratta di energia", l'elaboratore accetterebbe questa risposta, che in realtà è sbagliata.
Qui però si entra nel campo di una ricerca ben più vasta: una ricerca che coinvolge l'interpretazione dei testi scritti in linguaggio naturale e che supera l'ambito delle applicazioni dell'informatica alla didattica.

Problemi e prospettive

Un altro grosso problema è che il valore di questo metodo dipende dalla ricchezza delle ramificazioni del programma utilizzate per il ricupero degli errori.
Più rami ci sono, cioè più possibilità sono aperte per lo studente che sta imparando, e migliore sarà l'apprendimento.
Naturalmente è abbastanza evidente che esiste un limite all'ampliamento del programma: sia per il costo di produzione del programma stesso, sia eventualmente per la capacità di memoria della macchina.

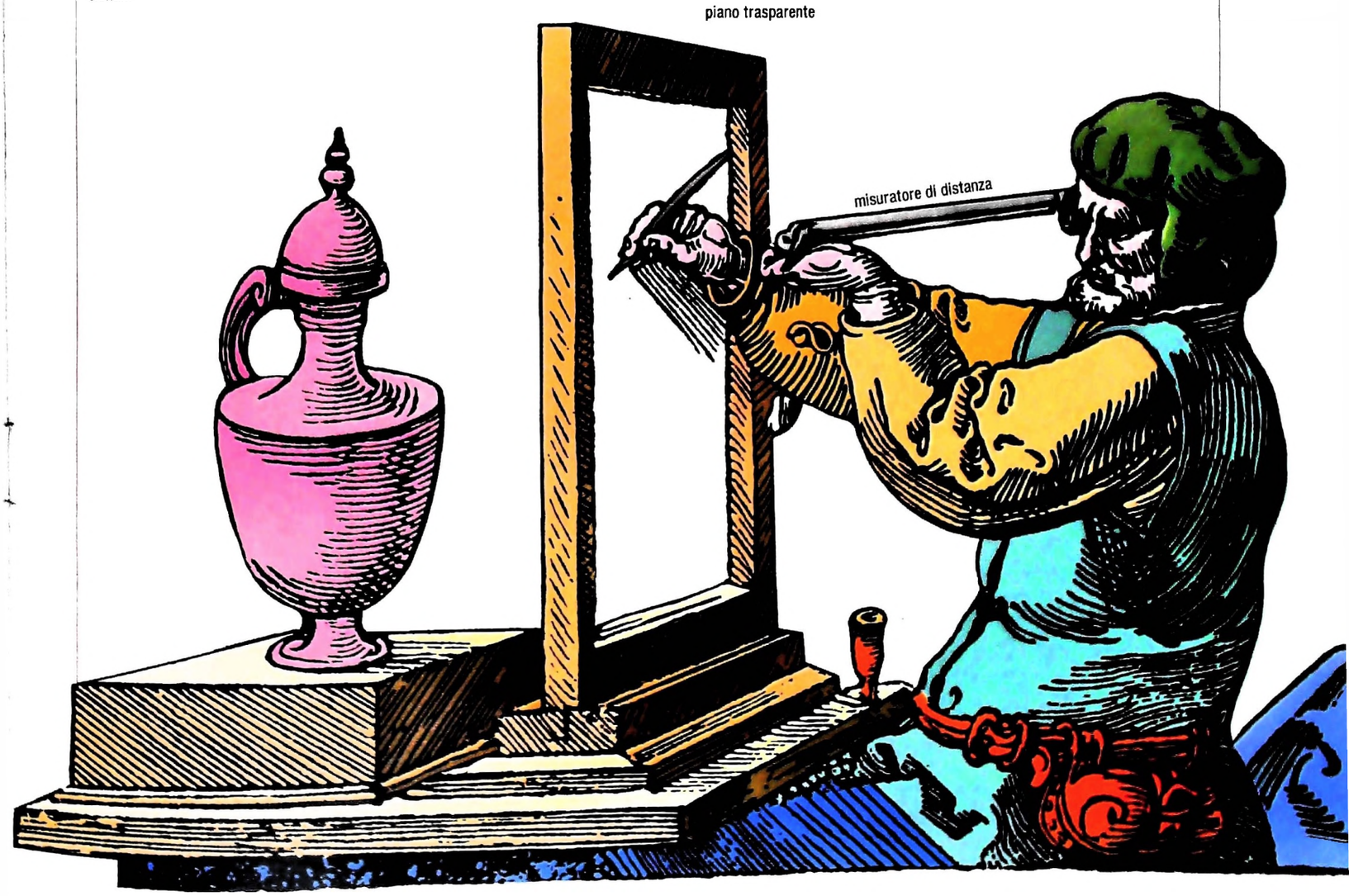
Una valida alternativa all'ampliamento eccessivo del sistema è la presenza di ramificazioni con vari livelli di difficoltà: l'elaboratore presenta l'uno o l'altro dei diversi rami a seconda del grado di preparazione mostrato dallo studente nelle fasi precedenti.
Si tratta di tentativi tuttora in fase di studio: la difficoltà consiste evidentemente nel trovare un metodo economico ed efficace per valutare il livello dello studente a seconda delle risposte da lui date.
Altri tentativi sono in corso in vista della costruzione di un programma che modifica, migliorandolo, il proprio stile di insegnamento a seconda dello studente che ha di fronte. Si tratta di programmi chiamati "self-improving" cioè che si "automigliorano". Ma anche in questo caso non è stata ancora superata la fase di ricerca e di sperimentazione.
Concludendo possiamo dire che il metodo *tutoriale* ha compiuto e sta compiendo notevoli progressi con risultati significativi, ma è ancora lontano dall'obiettivo di voler sfruttare pienamente le capacità dell'elaboratore e non essere soltanto un sostituto più o meno valido di un libro di Istruzione Programmata.

I PACCHETTI GRAFICI

Introduciamo la trasformazione di window, che ha lo scopo di trasformare le coordinate del mondo in coordinate dello schermo.

Sinora abbiamo imparato come ottenere dei semplici disegni sullo schermo dell'M10 e sul microplotter PL10. Vi sarete però accorti che un programma fatto per avere un output sullo schermo non è utilizzabile per ottenere lo stesso risultato sul plotter e viceversa. Ossia, la periferica di output non è indipendente dal programma. Per ovviare a questo inconveniente bisogna scrivere un certo numero di gruppi di istruzioni (chiamate subroutines) che costituiscano una sorta di filtro fra il programma e qualsiasi unità periferica di output, tipicamente schermo e plotter. L'insieme di queste subroutines si chiama "pacchetto grafico".
Un buon pacchetto grafico deve avere certe caratteristiche che ne garantiscano l'utilità. Alcune tipiche sono:
Semplicità. I programmi che costituiscono il pacchetto grafico devono essere ben commentati e le possibilità offerte (le funzioni grafiche) devono essere le più semplici possibile, per poter essere comprese rapidamente da ogni utente. Ogni pacchetto grafico deve inoltre essere corredato da un manuale d'uso.
Completezza. Le funzioni del pacchetto grafico devono essere, oltre che le più semplici possibile, anche complete.
In altre parole, devono costituire un insieme di poche operazioni grafiche elementari che permetta però di risolvere numerose classi di problemi.

Così Albrecht Dürer rappresentava il suo metodo per la riproduzione in prospettiva su un piano di oggetti tridimensionali.

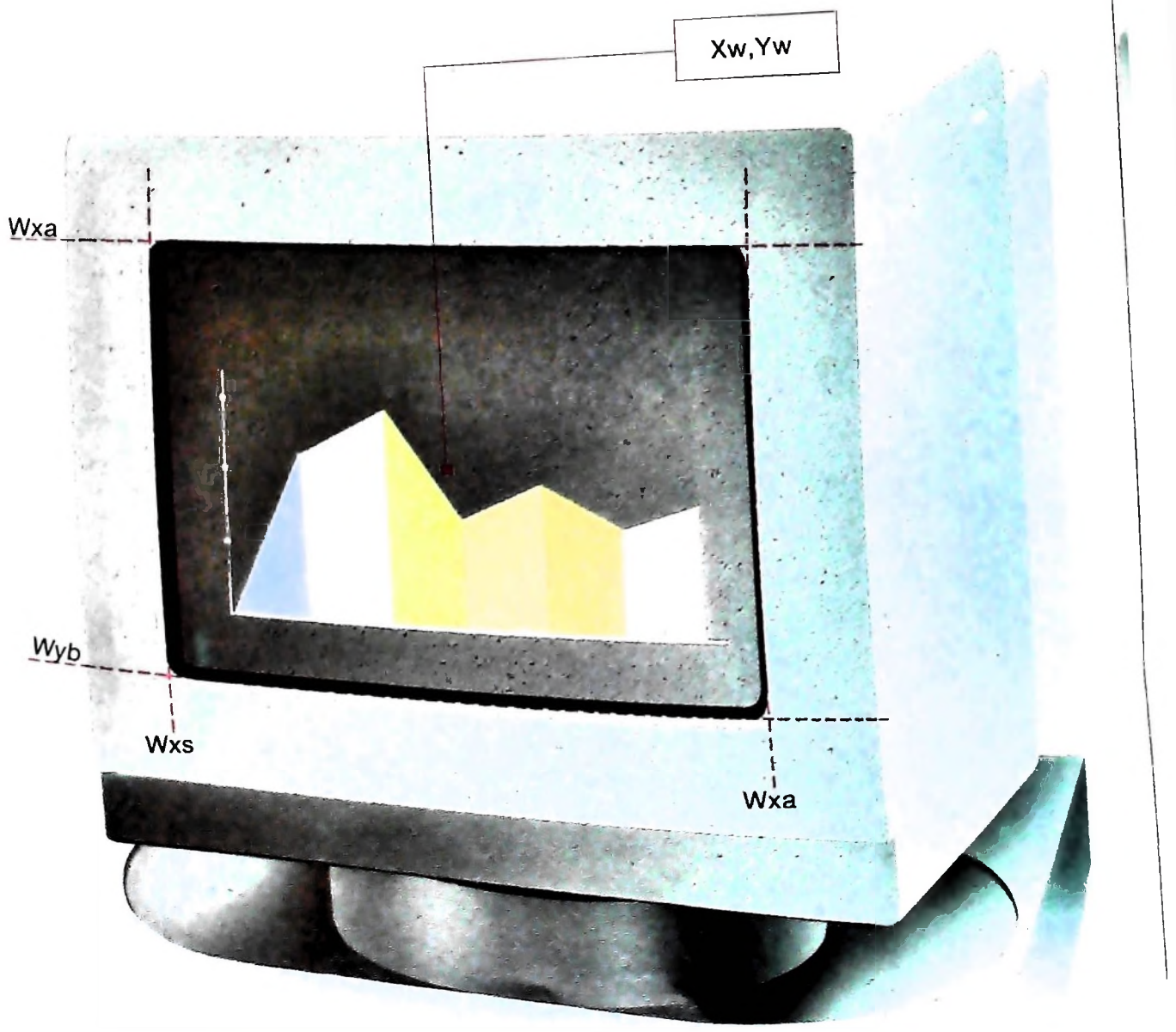


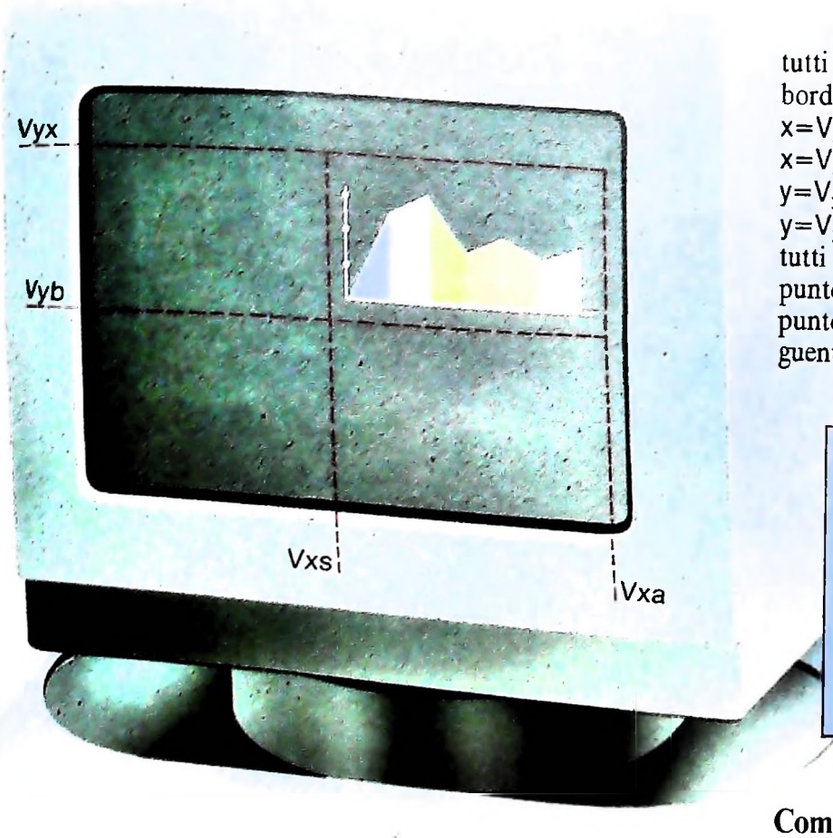
Robustezza. Il pacchetto grafico deve essere in grado di correggere automaticamente gli errori più semplici senza interrompere la sua esecuzione e senza dare segnalazioni. Solo per gli errori più strettamente di tipo logico deve interrompersi e segnalare l'errore individuato.
Economicità. Il pacchetto grafico deve essere piccolo ed economico, in termini di spazio di memoria utilizzato più ancora che di costo.
Prima di entrare più nel dettaglio della stesura di un pacchetto grafico, dobbiamo affrontare un altro problema. Supponiamo di dover disegnare un grafico, le cui dimensioni siano maggiori di quelle del nostro schermo. Come fare per riprodurlo? Ci si offrono due possibili scelte:
a) effettuare una trasformazione di scala, ovvero ridurre proporzionalmente le dimensioni del grafico in modo tale che sia riproducibile sullo schermo;
b) riprodurre sullo schermo solo la parte del grafico che ne può essere contenuta.

Il primo dei due possibili modi di procedere va sotto il nome di "trasformazione di window", mentre il secondo viene detto "clipping", dall'inglese "clip", taglio.
Vediamo ora di approfondire il discorso sulle trasformazioni di window; affronteremo invece il problema del clipping nelle prossime lezioni.

Window e viewport

Cominciamo col fissare alcuni concetti fondamentali della computer graphics. Il concetto di window è legato al modello teorico del disegno che dobbiamo rappresentare e che è definito in un sistema di riferimento assoluto, slegato dallo strumento computer, e tradizionalmente indicato come "sistema di coordinate del mondo". Aprire una window significa quindi definire un'apertura, una finestra sul mondo, di dimensioni specificate, che contenga il modello del nostro dise-





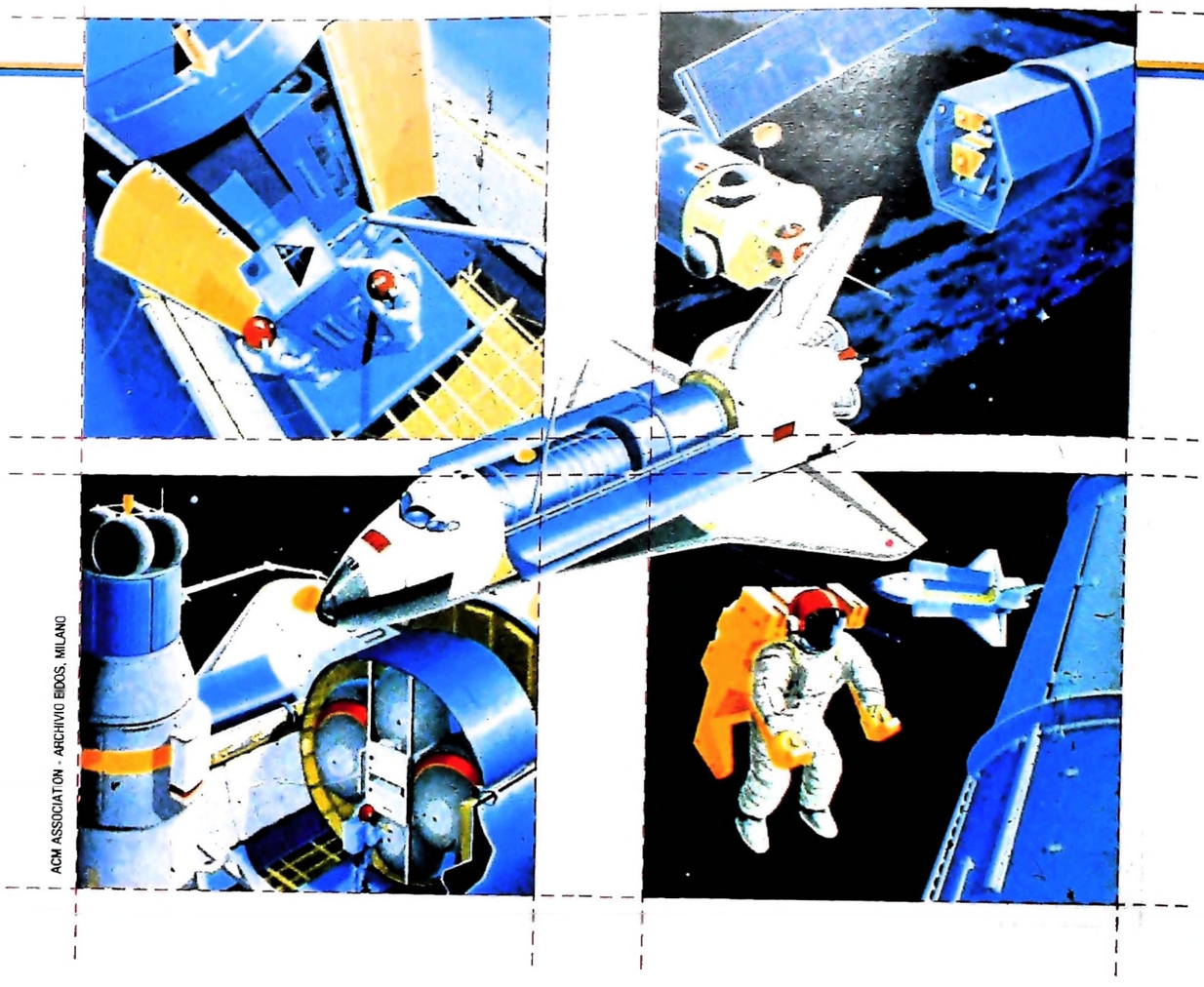
gno espresso in "coordinate del mondo". Per cui i vertici della window saranno anch'essi indicati utilizzando le coordinate del mondo.
Le trasformazioni di window hanno quindi lo scopo di convertire le coordinate del mondo in coordinate dello schermo. Ossia, trasportare un disegno contenuto in una window nel sistema di riferimento dello schermo del computer.
Possiamo inoltre far in modo che il contenuto della window venga riprodotto soltanto su una porzione dello schermo. Questa operazione viene indicata come "aprire una viewport" sullo schermo, ossia un "porto di arrivo per la window". La viewport è quindi definita mediante le coordinate dei suoi vertici espressi in coordinate dello schermo.
La figura seguente mostra una trasformazione di window da coordinate del mondo a coordinate dello schermo.
Riassumendo, usiamo la window per definire ciò che vogliamo disegnare, ed usiamo invece la viewport per specificare dove disegnare sullo schermo. Normalmente bisognerà avere l'accortezza di utilizzare window e viewport di forma simile, onde evitare distorsioni nella conversione del disegno dall'una all'altra.
Vediamo ora come si calcolano le coordinate dello schermo partendo da quelle del mondo.
Supponiamo che i bordi della window siano:
$x=W_{xs}$ (bordo sinistro);
$x=W_{xd}$ (bordo destro);
$y=W_{yb}$ (bordo in basso);
$y=W_{ya}$ (bordo in alto)
tutti espressi in coordinate del mondo, e che i corrispondenti bordi della viewport siano:
$x=V_{xs}$ (bordo sinistro);
$x=V_{xd}$ (bordo destro);
$y=V_{yb}$ (bordo in basso);
$y=V_{ya}$ (bordo in alto)
tutti espressi in coordinate dello schermo. Allora, il generico punto (X_w, Y_w) in coordinate del mondo, si trasforma nel punto (X_s, Y_s) in coordinate dello schermo, utilizzando le seguenti formule:

$$X_s = \frac{V_{xa} - V_{xs}}{W_{xa} - W_{xs}} (X_w - W_{xs}) + V_{xs}$$

$$Y_s = \frac{V_{ya} - V_{yb}}{W_{ya} - W_{yb}} (Y_w - W_{yb}) + V_{yb}$$

Commento alle formule

Il termine frazionario, per esempio $(V_{xa} - V_{xs}) / (W_{xa} - W_{xs})$ rappresenta il fattore di scala fra window e viewport, ed è un numero costante. In un programma può quindi essere utile calcolarselo a parte una prima volta, definirlo come costante e richiamare, quando risulta necessario, solo la costante stessa.
Il termine $(X_w - W_{xs})$ determina la distanza del punto (X_w, Y_w) dal bordo sinistro della window; l'addendo V_{xs} indica invece di quanto è traslato il bordo sinistro della viewport rispetto alle coordinate dello schermo, e quindi è una costante che va aggiunta nel calcolo dell'ascissa X_s.
Analoghe osservazioni valgono per Y_s.
Per convertire un disegno da una window a una viewport bisognerà quindi calcolare, per ogni punto espresso in coordinate del mondo, il rispettivo in coordinate dello schermo, mediante le formule precedenti.
Potreste provare, per esercizio, a definire una window sul mondo, disegnarvi all'interno una semplice figura costituita da segmenti e provare a scrivere un programma che riproduca il disegno sullo schermo, dentro la viewport che vi andrà di definire. Inoltre vi consigliamo di riguardare il programma che disegna il grafico della funzione sen(x), visto nella sesta lezione di grafica, nel quale è stata introdotta in maniera implicita una semplice trasformazione di window. Lo scopo, anche in quel caso, era di poter disegnare sullo schermo una figura, il grafico della funzione, che concettualmente era di dimensioni maggiori dello schermo stesso. La trasformazione di window era rappresentata dall'utilizzo delle costanti SCALA e TRASLA.
Nella prossima lezione vedremo alcune applicazioni sulle trasformazioni di window.



ACM ASSOCIATION - ARCHIVIO BIDOS, MILANO

Window di ieri, di oggi e di domani

La nozione di window, e la trasformazione cui vanno soggetti i dati numerici di una figura generata con un computer, ha origini che possono essere ricondotte molto indietro nel tempo. L'idea di Dürer di adottare uno schermo trasparente per realizzare un disegno prospettico (metodo che veniva chiamato Sportello di Dürer) è forse la più vicina. Secondo questo metodo veniva delimitato il campo di visione con uno schermo piano sul quale si procedeva a disegnare con una vera e propria tecnica di ricalco. La window è in questo caso l'area visibile attraverso lo sportello, che può essere più o meno ampia a seconda di quanto si è vicini allo schermo stesso. La viewport è lo stesso schermo su cui si forma l'immagine ricalcata.

Ritroviamo analoghe nozioni nella tecnica fotografica, con nomi diversi, ma possiamo sottolineare la corrispondenza che c'è tra window e area visibile, da una parte, e piano della pellicola e viewport dall'altra.

La forma della viewport è a sua volta molto importante. Nel mondo dei computer, parlando di terminali, si usa frequentemente la dizione formato paesaggio o formato ritratto, per distinguere schermi che si presentano nella loro forma rettangolare orientata orizzontalmente oppure verticalmente. Questa terminologia è derivata dalla pittura, in cui, come ricorda Kandinsky in "Punto, linea e superficie" si usa adottare un piano di disegno a sviluppo orizzontale per illustrare scene di paesaggi, mentre si adotta il formato verticale per drammatizzare una scena e in particolare per la ritrattistica. Questi sono ormai modi di percezione entrati nella nostra mente, e la grafica a computer tenta di emularli per facilitare la comunicazione con l'elaboratore.

Più recentemente è stata introdotta la nozione di "multiwindow"; si tratta di sistemi di elaborazione capaci di visualizzare l'informazione su uno schermo di un terminale suddividendolo in diverse zone. Si cerca con questa tecnica di simulare quello che normalmente facciamo sul tavolo da lavoro, in cui usiamo diversi fogli su cui scriviamo, prendiamo appunti, annotiamo appuntamenti. Di volta in volta un foglio viene portato in superficie e si lavora su di esso, poi viene coperto da un altro, ma in ogni momento possiamo ripescarlo e correggere quanto vi è scritto, controllare un dato, verificare un impegno di lavoro. Lo scopo di questi sistemi multiwindow è appunto quello di mettere a disposizione dei singoli utenti un ambiente che assomigli il più possibile a quello usuale, ma, mentre nella realtà il passaggio da un foglio a un altro avviene sollevando e sovrapponendo materialmente, in questi casi si sceglie un foglio indicando con il cosiddetto "mouse" su quale area si vuole operare.

Le window in questi casi sono dunque vere e proprie finestre sui diversi mondi in cui si opera lavorando: l'agenda, il foglietto di appunti, la lettera che stiamo scrivendo.

PERSONAL COMPUTER OLIVETTI M10

L'UFFICIO DA VIAGGIO

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 collegarsi via telefono per spedire o ricevere informazioni.

Qualunque professione sia la vostra, M10 è in grado, dovunque vi troviate, di offrirvi delle capacità di soluzione davvero molto grandi. M10: il più piccolo di una grande famiglia di personal.

olivetti

Per informazioni rivolgersi ai 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

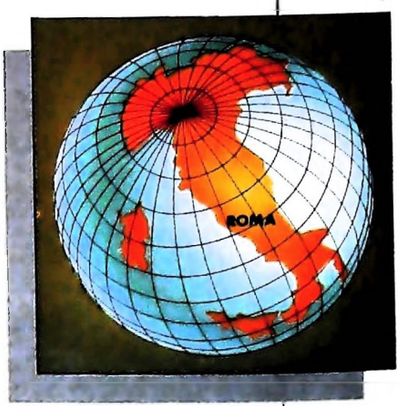
Anche in leasing con Olivetti Leasing

Banca Aperta

LE NUOVE RISPOSTE DEL BANCO DI ROMA.

Vorrei avere un rapporto più diretto con la mia banca...

Anche le strutture bancarie si evolvono. Il Banco di Roma, primo in Italia, sta introducendo la struttura a "banca aperta", già attuata da molte sue filiali italiane. "Banca aperta": non il solito bancone, le lunghe file, ma un nuovo modo di essere banca, un rapporto più personalizzato, un clima più agevole, più professionale e una maggiore rapidità in ogni operazione. Un ulteriore passo avanti verso la completa consulenza finanziaria che il Banco di Roma intende mettere a disposizione dei propri clienti. Tra i numerosi servizi offerti ricordiamo: Prestito Personale, Prestito Casa, gestione dei patrimoni, Leasing, assistenza all'import-export, attraverso ben 60 sedi estere in 30 Paesi dei 5 continenti. Tutto questo perché il Gruppo Banco di Roma è in grado di gestire ogni servizio specifico con grande professionalità, fornendo anche informazioni dirette a domicilio attraverso i sistemi Videotel e Voxintesi.



BANCO DI ROMA

CONOSCIAMOCI MEGLIO.