CADEL

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

21 CORSO PRATICO COL COMPUTER

421735

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

F4 F5 F6 F7 F8

diretto da GIANNI DEGLI ANTONI

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

FABBRI EDITORI

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

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

INTERNO©

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

CPU - C-MOS da 8 bit
Permette un'alta velocità d'elaborazione con poco consumo d'energia.

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

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

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

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

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

Accessori opzionali

L'interfaccia registratore/stampante grafica CE-150
- Come stampante realizza l'hardcopy di programmi, dati e calcoli e offre una funzione grafica a quattro colori: rosso, nero, verde, blu. Stampa i caratteri in nove corpi diversi. Ogni riga ne può contenere da 4 a 36. Questa stampante può essere controllata dall'operatore orientando il senso di stampa verso l'alto, il basso, a destra e a sinistra.
- Come interfaccia si può collegare ad uno o due registratori (uno per la memorizzazione dei dati e programmi, l'altro per richiamarli).

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

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

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

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

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

concessionaria per l'Italia
MELCHIONI

TUTTA LA POTENZA DI UN COMPUTER NEL PALMO DELLA TUA MANO

Per ulteriori informazioni scrivete a
MELCHIONI - Divisione Pocket Computer - 20135 MILANO - Via P. Colletta 37

SHARP

Direttore dell'opera
GIANNI DEGLI ANTONI

Comitato Scientifico
GIANNI DEGLI ANTONI
Docente di Teoria dell'Informazione, Direttore dell'Istituto di Cibernetica dell'Università degli Studi di Milano

UMBERTO ECO
Ordinario di Semiotica presso l'Università di Bologna

MARIO ITALIANI
Ordinario di Teoria e Applicazione delle Macchine Calcolatrici presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

MARCO MAIOCCHI
Professore incaricato di Teoria e Applicazione delle Macchine Calcolatrici presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

DANIELE MARINI
Ricercatore universitario presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

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

Testi
ENRICO BORELLO, ADRIANO DE LUCA, GOFFREDO HAUS, Etnoteam (ADRIANA BICEGO), GIOVANNI ROCCA

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 BREGGÉ

Segretarie di Redazione
RENATA FRIGOLI
LUCIA MONTANARI

L'UAMICRO I: PARTE III

Prosegue l'esame dei componenti funzionali del microprocessore UAMICRO I.

Registro "O" (uscita)

Il registro "O" (OUTPUT) è un registro di uscita che può avere molti usi: nel nostro caso è soltanto una piccola memoria (BUFFER) dove viene immagazzinato un dato, una istruzione o semplicemente il risultato di una operazione per passarlo al nostro visualizzatore DISPLAY. Un tipo di visualizzatore molto usato oggi è il ben conosciuto televisore, ma ce ne sono altri, come sistemi di stampa, grafica e alfanumerici luminosi ecc. Quello che comunque tutti hanno più o meno in comune sono i registri di immagazzinamento.

Istruzioni

Le istruzioni sono il linguaggio dei microprocessori. Infatti sono l'unico mezzo che ci permette di programmarli. La quantità come la qualità delle istruzioni dipende dal microprocessore, però possiamo raggrupparle per funzioni secondo la suddivisione seguente:

— ISTRUZIONI DI TRASFERIMENTO
— ISTRUZIONI DI ENTRATA E USCITA DATI
— ISTRUZIONI DI ARITMETICA E LOGICA
— ISTRUZIONI DI ROTAZIONE E SCORRIMENTO
— ISTRUZIONI DI MANIPOLAZIONE DI BIT
— ISTRUZIONI DI SALTO, CHIAMATA A SOTTOPROGRAMMA E RITORNO

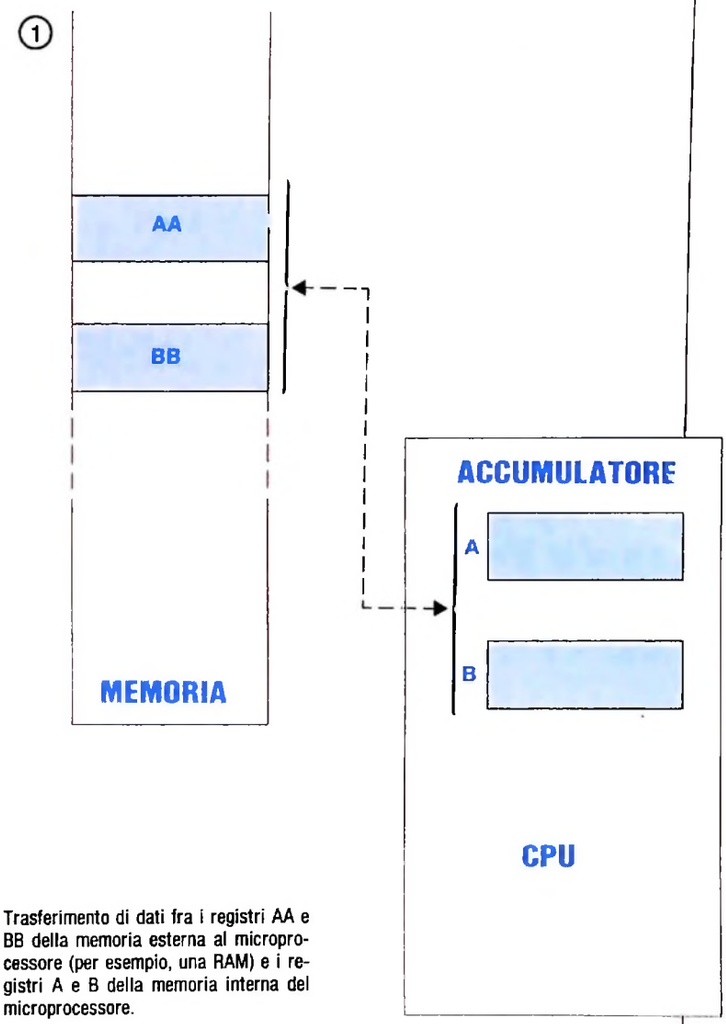
In realtà, le funzioni qui sopra possono essere ridotte ancora a tre sole secondo una struttura di più alto livello:

— ISTRUZIONI PER TRASFERIMENTO DATI
— ISTRUZIONI PER OPERAZIONI
— ISTRUZIONI DI CONTROLLO

Naturalmente, per rendere più chiara la nostra analisi, prenderemo in considerazione solo la prima suddivisione.

Istruzioni di trasferimento di dati

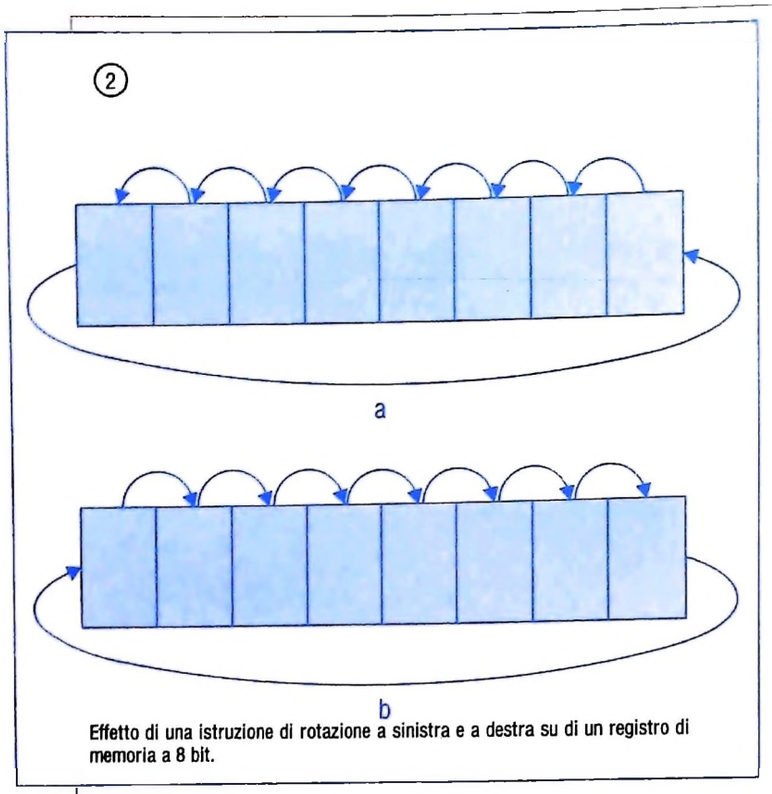
Queste istruzioni sono le più comuni: esse ci permettono di trasferire dati da una locazione di memoria a un'altra oppure all'interno della CPU (figura 1).
In definitiva, esse si incaricano esclusivamente del trasferimento dei dati sia internamente che esternamente ai sistemi costruiti con i microprocessori.



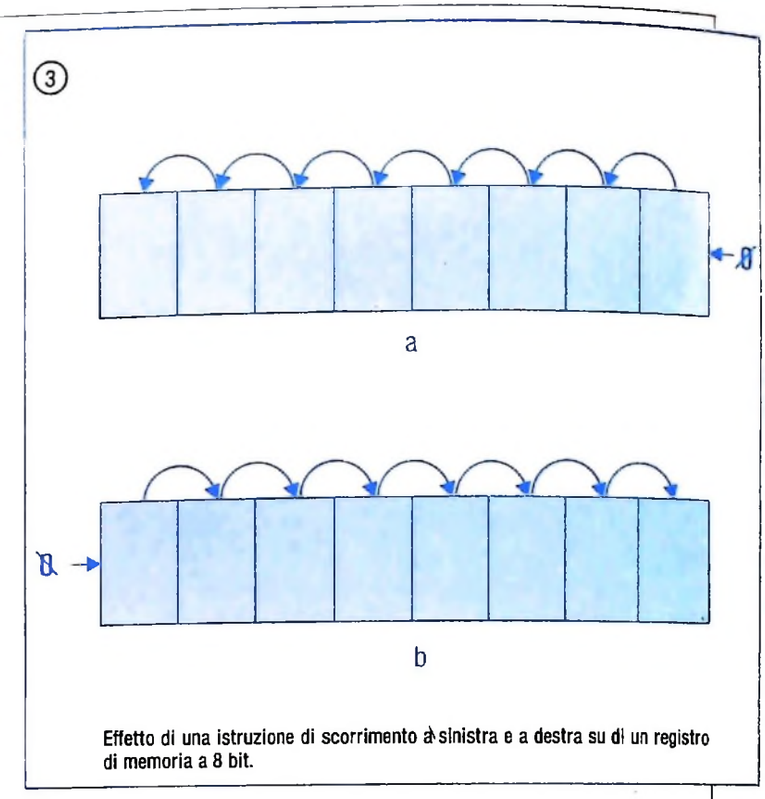
Trasferimento di dati fra i registri AA e BB della memoria esterna al microprocessore (per esempio, una RAM) e i registri A e B della memoria interna del microprocessore.

Istruzioni di entrata e uscita dati

Queste istruzioni, come abbiamo già avuto occasione di imparare, sono istruzioni che permettono l'entrata e l'uscita da o verso l'esterno del microprocessore, come per esempio un terminale Video, un altro calcolatore esterno oppure a sistemi di controllo di processi ecc.
Questi tipi di istruzioni, in ogni modo, sono presenti solo in microprocessori del tipo Von Neumann completo mentre per quelli con Von Neumann ridotto valgono le istruzioni di trasferimento dati spiegate in precedenza.



Effetto di una istruzione di rotazione a sinistra e a destra su di un registro di memoria a 8 bit.



Effetto di una istruzione di scorrimento a sinistra e a destra su di un registro di memoria a 8 bit.

Istruzioni di aritmetica e logica

Le istruzioni aritmetiche eseguono operazioni di somma e sottrazione, moltiplicazione e divisione (queste due ultime solo con microprocessori a 16 bit) sui dati immagazzinati in memoria o comunque presenti nella CPU, mentre le istruzioni logiche eseguono operazioni di AND, OR, INV. e OR ESCLUSIVO.

Istruzioni di rotazione e scorrimento

Queste istruzioni agiscono sempre sul contenuto dei registri interni del microprocessore. Le prime trasferiscono (figura 2) l'ultimo bit, sia quello a destra o a sinistra, secondo il tipo di istruzioni scalando tutti gli altri di un posto; questa operazione può essere ripetuta molte volte. Le istruzioni di scorrimento (figura 3), invece, fanno scorrere il dato verso destra o sinistra di una o più locazioni, però i bit che escono fuori dal registro vanno persi, mentre nei posti che restano vuoti, dalla parte opposta, entrano degli zeri. Se provate a mettere un dato e lo fate scorrere a destra o a sinistra inserendo zeri vedrete che il valore del dato viene moltiplicato o diviso per due a ogni scorrimento.

Istruzioni di manipolazione di bit

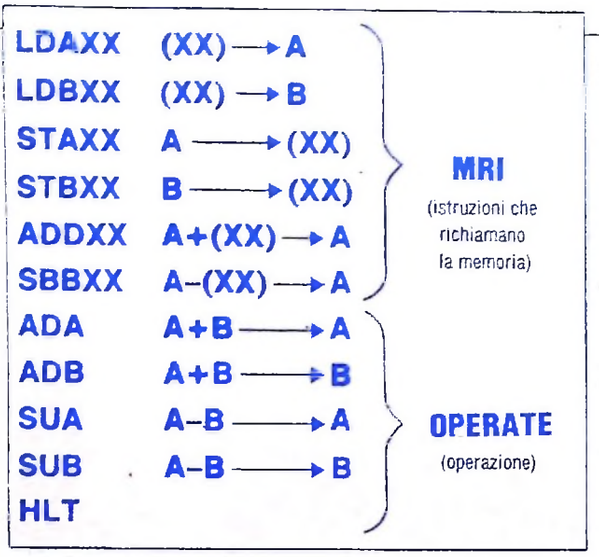
Queste istruzioni permettono di controllare il valore di un determinato bit di un dato sia in memoria sia nei registri interni del microprocessore.

Istruzioni di salto, chiamata a sottoprogramma e ritorno

Le istruzioni di salto sono già state trattate e sono le istruzioni che ci permettono di saltare da un punto a un altro del programma secondo un determinato schema. Le istruzioni di chiamata a sottoprogramma invece sono più complesse e importanti, anche perché il loro uso è aumentato enormemente a seguito delle nuove organizzazioni dei linguaggi moderni. La loro trattazione verrà fatta approfondendo maggiormente i dettagli più avanti quando analizzeremo l'UAMICRO II.

La struttura delle istruzioni

Dopo aver spiegato le funzioni dei vari gruppi di istruzioni passiamo ora a trattare la loro struttura. Le istruzioni vengono date al sistema sotto forma di codice mnemonico, per due motivi: perché in genere il codice mnemonico che è una contrazione della parola che esprime l'azione della istruzione, è più facile da ricordare; inoltre perché si riducono gli errori di programmazione dovuti al fatto che è molto difficile trattare direttamente con i codici in forma numerica o addirittura con le cifre 0 e 1. In ogni modo, il microprocessore riesce a leggere solo 0 e 1, per cui ci deve essere qualche altra cosa, cioè un software speciale, che trasforma i codici mnemonici in codice binario e questo software speciale si chiama linguaggio assemblatore o "assembler". Esso varia da un microprocessore ad un altro, in quanto dipende dalla sua struttura hardware.



④

Rappresentazione grafica delle istruzioni del microprocessore UAMICRO I. Codice mnemonico (a sinistra), codice operativo (al centro) e sua rappresentazione esadecimale (a destra) delle operazioni dell'UAMICRO I.

⑤

CODICE MNEMONICO	CODICE OPERATIVO	
LDA	00000000	00H
LDB	00000001	01H
STA	00000010	02H
STB	00000011	03H
ADD	00000100	04H
SBB	00000101	05H
ADA	11110000	F0H
ADB	11110001	F1H
SUA	11110010	F2H
SUB	11110011	F3H
HLT	11110110	F6H

Istruzioni della UAMICRO I

L'UAMICRO possiede solo 12 istruzioni (in realtà la quantità adesso non ha importanza) divise in due gruppi: quelle in cui l'operando si trova in memoria vengono dette istruzioni MRI; quelle che agiscono sui registri interni dei microprocessori vengono chiamate Operazioni (OPERATE) (figura 4).
Vediamo cosa vogliono dire:

LDA XX (XX)→A prima di tutto con XX si indica un numero generico (un indirizzo nel nostro caso) che va da 00H a FFH.

Con la forma (XX) si indica il contenuto della locazione di memoria con indirizzo XX.

L'istruzione LDAXX vuol dire "caricare nell'accumulatore A il contenuto della memoria con indirizzo XX".

LDB XX: caricare l'accumulatore B con il contenuto della memoria con indirizzo XX.

STA XX: caricare il contenuto dell'accumulatore A nella locazione di memoria con indirizzo XX.

STB XX: caricare il contenuto dell'accumulatore B nella locazione di memoria con indirizzo XX.

ADD XX: sommare il contenuto dell'accumulatore A con il contenuto della memoria con indirizzo XX e depositare il risultato in A (cancellando il valore dell'addendo che c'era prima).

SBB XX: come sopra, solo che è una sottrazione.

Finora abbiamo visto le istruzioni MRI che si caratterizzano per il fatto che come operando hanno un indirizzo di memoria.
Vediamo ora le istruzioni OPERATE.

ADA: sommare il contenuto dell'accumulatore A con quello di B e depositare la somma nell'accumulatore A.

ADB: uguale a quanto sopra, tranne che l'accumulatore finale è il B.

SUA: sottrarre B da A e depositare la differenza in A.

SUB: sottrarre B da A e depositare la differenza in B.

HLT: ferma qualsiasi operazione.

Microprogrammazione

Definita la quantità di istruzioni necessarie e come funzionano, dobbiamo assegnare ad ognuna di esse un codice. Le tecniche di assegnazione variano da costruttore a costruttore; noi useremo la tecnica della individuazione per gruppi (figura 5), cioè se i primi quattro bit, a partire da sinistra, sono zeri, allora vuol dire che sono istruzioni MRI, se invece sono uno allora sono OPERAZIONI. Nella figura, la colonna a sinistra rappresenta i codici mnemonici, la colonna interna la rappresentazione binaria e quella a destra l'esadecimale.

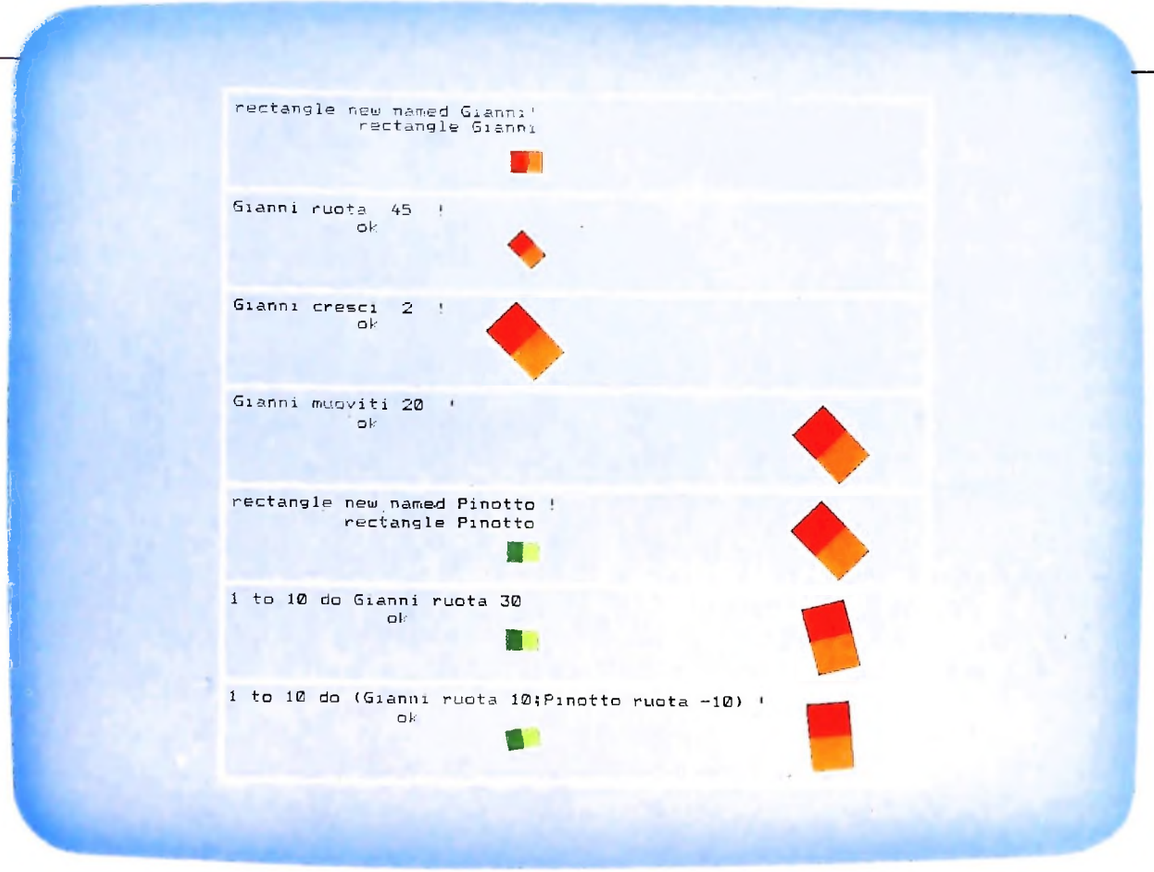
SULLA COMUNICAZIONE

Come modificare gli schemi di comportamento del computer per renderli più simili a quelli umani.

Un problema che si incontra nell'utilizzo di un calcolatore è quello della comunicazione di concetti e di informazioni fra l'utente e la macchina stessa. È noto come la comunicazione sia più semplice fra persone che hanno schemi mentali di ragionamento e di organizzazione delle informazioni simili. Il meccanismo del pensiero umano si è evoluto nel corso di millenni e difficilmente subirà cambiamenti radicali in futuro: è quindi più ragionevole, per permettere una più semplice comunicazione uomo-macchina, modificare gli schemi di comportamento dei computer per renderli più simili a quelli umani che viceversa. In alcuni articoli precedenti si è visto come gli sviluppi della ricerca nel campo dell'Intelligenza Artificiale abbiano portato alla comprensione di come l'informazione venga percepita e ricordata, e si è analizzato un tipo di comunicazione, quello iconico, che mette in grado l'utente di avere un rapporto più amichevole e comprensibile con il computer. In questo articolo vedremo quali approcci e quali ambienti di programmazione si sono sviluppati per avvicinare l'organizzazione delle informazioni all'interno di un computer e la presentazione di queste all'organizzazione della mente umana; analizzeremo anche alcune macchine (calcolatori) che utilizzano approcci di questo tipo.

La comunicazione ad oggetti

Nell'esperienza comune, la comprensione della realtà richiede la partecipazione a essa. Durante questo processo si tende a fare delle distinzioni e delle classificazioni per identificare elementi unitari all'interno della massa degli oggetti che costituiscono l'universo dell'esperienza. L'identificazione di un oggetto, con le sue proprietà associate (un quadrato ha come proprietà la sua forma, il suo colore, la sua posizione...) e la sua classificazione all'interno di un gruppo è dunque un elemento fondamentale nel modo di pensare e di percepire la realtà, e dovrebbe essere quindi presente in un sistema di programmazione evoluto. Nel seguito vedremo come questa caratteristica sia stata introdotta all'interno dei sistemi di programmazione. I sistemi software sono generalmente visti come composti di due entità distinte: una collezione di dati, le informazioni, e una collezione di procedure che manipolano queste informazioni. In una visione di questo tipo, i dati e le procedure costituiscono dunque entità ben distinte, anche se in effetti non lo sono. Una procedura infatti, quando viene invocata, fa delle assunzioni sulla struttura dei dati che le vengono passati come argomenti: sono ben note le difficoltà che incontrano coloro che cercano di applicare procedure a dati con struttura diversa da quella prevista al momento della definizione della procedura stessa. La scelta inoltre di dati e procedure appropriate è lasciata al programmatore. Questi due tipi di problemi sono stati individuati e parzialmente risolti, in un contesto in cui dati e procedure siano entità distinte, attraverso la tipizzazione dei dati, grazie alla quale il programmatore è avvertito se cerca di applicare procedure a tipi di dati diversi da quelli che queste si aspettano. Al contrario in un *sistema di programmazione ad oggetti*, ma conformemente all'esperienza comune, le due entità, dati e procedure, sono conglobate in un'unica struttura che le rappresenta entrambe. Come i dati, anche gli oggetti possono essere manipolati ed essi stessi possiedono le procedure che sono in grado di farlo. Per manipolare un oggetto gli si manda un messaggio che specifica quale procedura debba essere applicata all'oggetto stesso. Questo è il punto fondamentale e di grande rilievo che caratterizza la programmazione in un sistema ad oggetti: il messaggio, e in particolare il selettore all'interno del messaggio, specifica solamente quale azione debba essere compiuta sul ricevitore del messaggio stesso e non il modo in cui questa azione debba essere eseguita. La conoscenza del come l'azione debba essere espletata è conglobata nell'oggetto stesso che riceve il messaggio e non è al suo esterno. Un esempio di ciò può essere visto nella figura in alto della pagina a lato. Il programmatore, in un sistema ad oggetti, manda un messaggio che indica un tipo di manipolazione, non chiama una procedura: un messaggio nomina una manipolazione, non la descrive. Sia i messaggi che le procedure hanno un nome; ma un nome, in una visione tradizionale, indica una procedura specifica con il suo comportamento definito associato, mentre in un sistema a oggetti esso indica un tipo di manipolazione che può essere diverso a seconda dell'oggetto che la riceve (come abbiamo visto la conoscenza di come rispondere a un messaggio è conglobata *all'interno* degli oggetti stessi). Così se mandiamo il messaggio *muoviti* ad un oggetto diverso da un rettangolo, per esempio a un triangolo, otterremo il risultato di muovere il triangolo senza conoscere nulla sulla struttura particolare (lati, misura, colore...) dell'oggetto triangolo. Un messaggio, oltre al selettore, può contenere altri oggetti che prendono parte alla manipolazione: questi sono gli argomenti del messaggio. Nel messaggio *muoviti...*, ad esempio, esiste un solo argomento che specifica di quanto un oggetto debba muoversi. La descrizione di un tipo di manipolazione è fatta in una entità che è detta *metodo*. Esso, come una procedura, descrive le



Un esempio elementare di utilizzo dello Smalltalk. Si creano dei rettangoli e si fanno muovere e ruotare sullo schermo.

```
-----------------------------------------------
Class name                      Point

Superclass                      Object

Instance variables names        x y
-----------------------------------------------
class messages and methods

      newX: xValue Y: yValue ||
            ^ self new x: xValue
                       y: yValue
-----------------------------------------------
instance messages and methods

      X || ^x
      Y || ^y

      x: Xcoord  y: Ycoord ||
          x <- Xcoord
          y <- Ycoord

      + aPoint ||
          ^Point newX: x + aPoint X
                     Y: y + aPoint Y

      - aPoint ||
          ^Point newX: x - aPoint X
                     Y: y - aPoint Y
-----------------------------------------------
```

Un esempio di Smalltalk per la definizione di una classe, la Point (Punto). Vi sono definiti metodi per creare istanze della classe Punto, accedere alle coordinate x e y, per assegnare a queste nuovi valori e sommare e sottrarre alle coordinate di una istanza quelle di un altro punto.

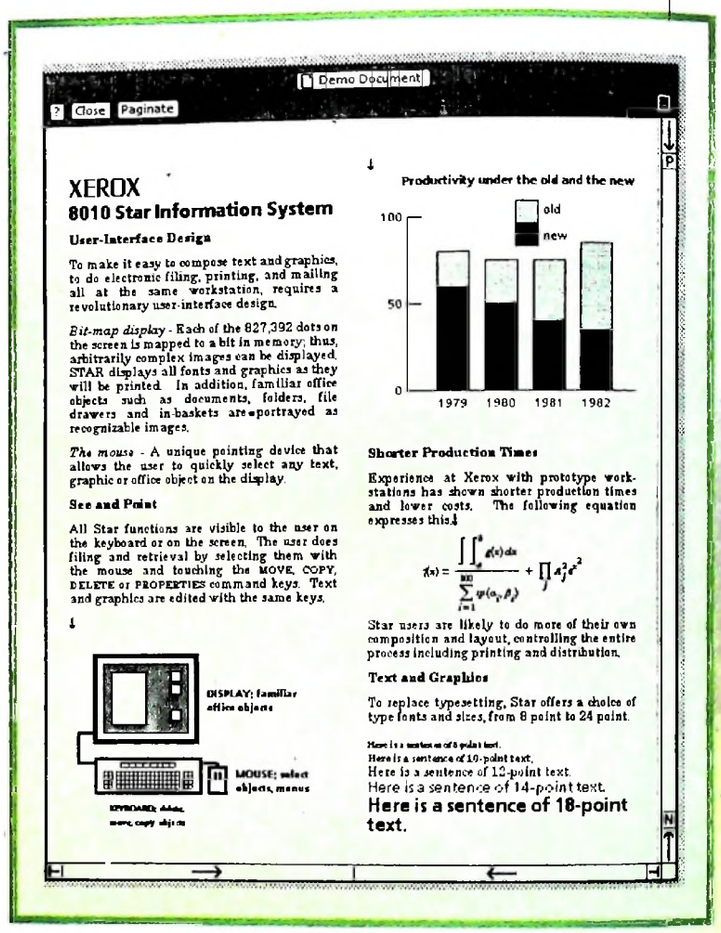
operazioni da effettuare su un oggetto, ma non può essere separato dall'oggetto stesso e non può richiamare procedure, ma soltanto spedire messaggi. Gli oggetti appaiono in modo diverso se guardati dall'esterno o dall'interno: dall'esterno appaiono come entità cui è possibile chiedere di eseguire un'azione (attraverso un messaggio), mentre dall'interno, cioè dal punto di vista del programmatore, è visibile il modo che ha l'oggetto di rispondere quando è sollecitato da un messaggio (attraverso il metodo associato al messaggio). I messaggi a cui un oggetto è in grado di rispondere sono detti il suo *protocollo* ed è questo protocollo che definisce le interazioni possibili dell'oggetto con l'esterno. All'interno dell'oggetto, come abbiamo detto, esistono anche dei dati che specificano le sue proprietà fisiche quali, per i nostri rettangoli, la posizione, la dimensione e così via: queste sono le *variabili* private dell'oggetto. Un oggetto singolo, in generale, viene classificato, nell'esperienza comune, come appartenente a un gruppo di cui condivide le caratteristiche di base: un oggetto, cioè, è un esemplare di una classe di oggetti simili (un rettangolo ha sempre le stesse caratteristiche: se noi prendiamo due rettangoli essi condividono il fatto di avere quattro lati, di avere, compresi fra questi, angoli di novanta gradi, di poter essere mossi e così via). In un ambiente di programmazione a oggetti, un gruppo di questi con le medesime caratteri-

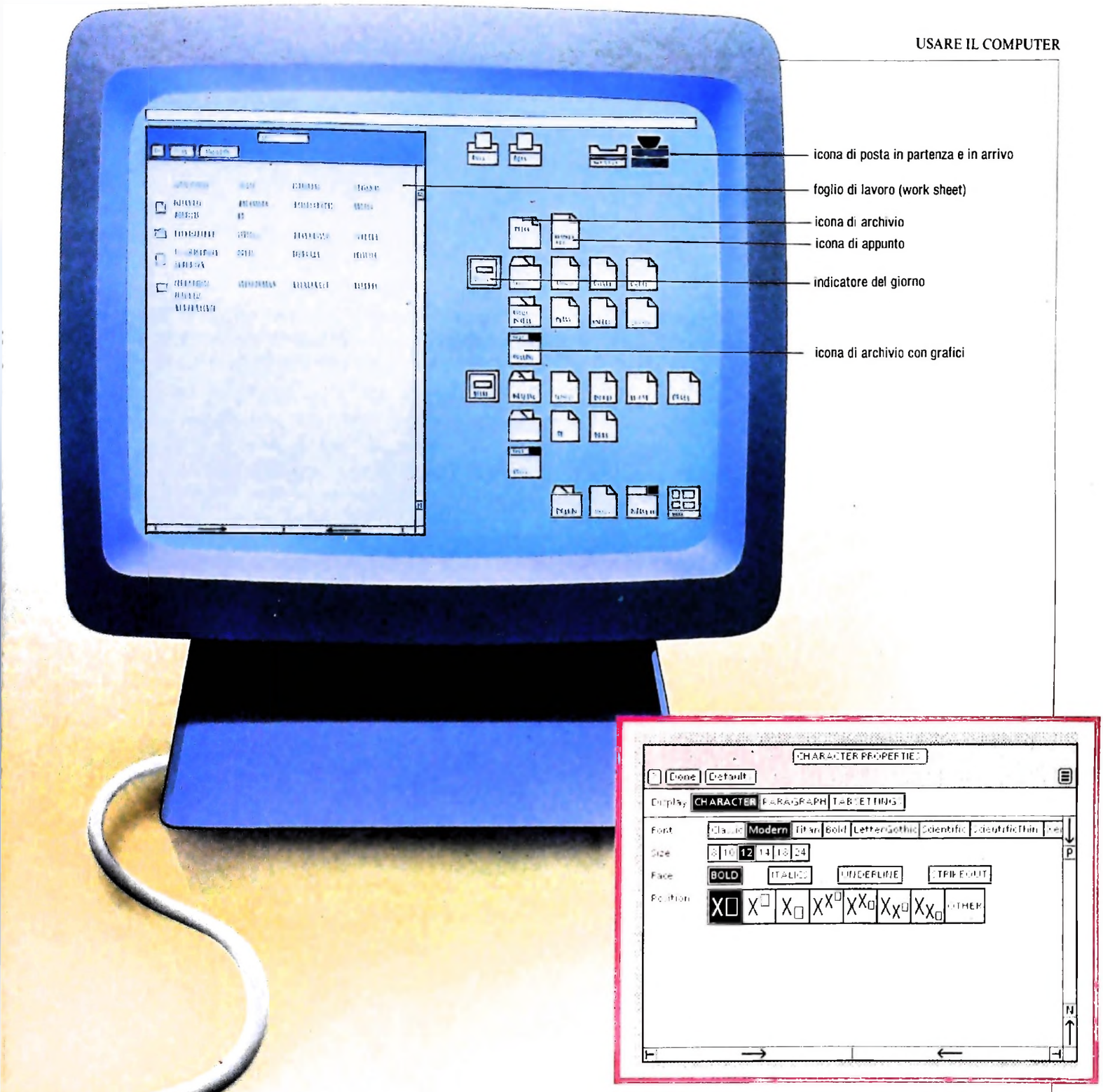
stiche è detto *classe*; un elemento singolo di questa classe è detto *istanza* della classe. Nella figura in basso della pagina precedente è mostrato un esempio in cui si definisce la classe Punto con i metodi per accedere alle sue variabili e manipolare quindi istanze di questa classe. Un'ultima, ma non meno importante caratteristica di un sistema a oggetti, è quella dell'*eredità*. Un oggetto può ereditare, se desiderato, gli attributi che caratterizzano un altro oggetto; ad esempio, se si vuole definire la classe "rettangolo con didascalia", i metodi e le variabili per trattare la parte geometrica verranno presi (ereditati) dalla classe rettangolo, evitando così di doverli ridefinire e di sprecare inutilmente spazio di memoria, che è una risorsa ancora limitata negli attuali computer.

Le macchine "moderne"

A partire dai concetti sopra esposti, che sono stati sviluppati presso il PARC (Palo Alto Research Center) della Xerox negli anni Settanta, e utilizzando a fondo il concetto di icona, sono state sviluppate alcune macchine che consentono un rapporto con l'utente radicalmente diverso e molto più amichevole di quello tradizionalmente disponibile sui computer. La prima di queste macchine, che ha tracciato la via secondo cui muoversi, è stata la Star della Xerox. All'utente viene presentata una "videata" (desktop) che attraverso l'uso di icone evocative rappresenta la scrivania che di solito si usa. L'interazione con la macchina, come in ogni altro sistema che si avvale delle icone, avviene attraverso il "mouse" che viene utilizzato per puntare una posizione particolare sullo schermo: il cursore sullo schermo è infatti connesso al mouse e ogni spostamento di questo sul piano della scrivania si ripercuote equivalentemente sul cursore stesso. Sul mouse esistono dei pulsanti con cui si selezionano gli oggetti precedentemente puntati sullo schermo. Ora, sullo Star, le icone rappresentano gli oggetti normalmente disponibili su una scrivania (documenti, cartellette, archivi, contenitori per la posta in arrivo e in partenza, e così via). L'interazione con questi oggetti avviene spostandosi sulla icona che li rappresenta e selezionandola con il mouse, premendo il pulsante O, come si suole ormai dire, "clickando". A questo punto, l'icona, che rappresenta l'oggetto selezionato, si espande in una finestra, uno schermo virtuale attraverso cui avviene poi l'interazione con l'oggetto stesso: ad esempio, un documento viene espan-

Lo STAR 8010 della Xerox: un sistema per "office automation" dotato di elevate capacità grafiche e caratteristiche di interfaccia uomo-macchina molto avanzate e di tipo orientato all'utente ("user friendly").
Nella pagina accanto, in alto, compare il suo terminale video, che l'utente sfrutta adoperando un puntatore pilotato dal "mouse" (topolino), la scatolina in basso, senza fare ricorso ad alcun linguaggio di programmazione.
Sul video compaiono simboli grafici ("icone"), che rappresentano sia gli oggetti che si vogliono adoperare (lettere, cartellette, archivi, contenitori) sia le azioni che si vogliono compiere su tali oggetti. In basso, come si imposta la stampa di un testo tipografico, definendone formato, corpi, carattere ecc.





so come nella figura in alto a sinistra. Il fatto di selezionare un oggetto e di clickare su esso corrisponde in realtà, essendo questo un ambiente a oggetti del tipo visto in precedenza, a inviare all'oggetto selezionato il messaggio "mostrati". Abbiamo visto che un oggetto ha la conoscenza di come rispondere a un messaggio e quindi l'inviare lo stesso tipo di messaggio a oggetti (icone che li rappresentano) diversi ottiene l'esposizione degli stessi con caratteristiche che dipendono dalla loro rappresentazione interna. È proprio grazie a questa caratteristica che, nello Star, il set di comandi disponibili è limitato a: move, copy, show properties, copy properties, again, undo ed help, e si noti bene che questo non costituisce affatto una limitazione. Tali comandi hanno una corrispondenza diretta con le caratteristiche di un sistema a

Icone di base del sistema Macintosh della Apple Inc.

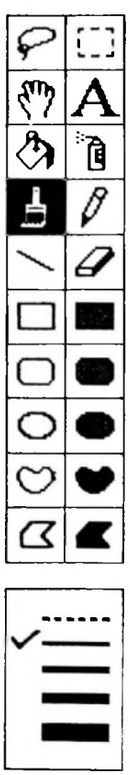
oggetti esposte in precedenza. Il comando **copy** crea in pratica un'istanza di un oggetto, il comando **show properties** rende visibile la struttura interna di un oggetto, **copy properties** fa in modo che un oggetto erediti le proprietà di un altro oggetto. Un esempio di come sono rappresentate le caratteristiche di un oggetto è nella figura in basso della pagina precedente, dove si possono vedere e selezionare gli attributi di un carattere all'interno di un documento. Si può definire la fonte utilizzata, il corpo del carattere e la sua posizione su una riga.

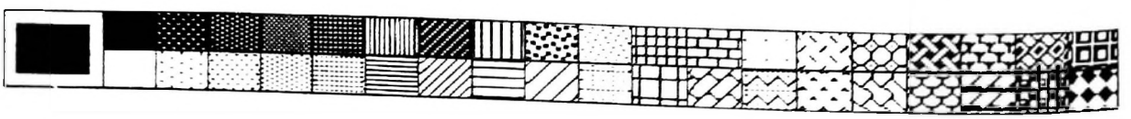
Altre macchine che utilizzano icone, anche se non seguendo una metodologia raffinata ed uniforme come quella dello Star, sono, per esempio, quelle della famiglia a trentadue bit della Apple: **Lisa e Macintosh**. In esse l'interazione avviene sempre utilizzando il mouse e le finestre attraverso la selezione di icone. Le icone di base utilizzate sono quelle mostrate nella figura sopra, per il Macintosh: clickando su di esse si apre l'applicazione corrispondente, senza dover scrivere alcun comando sulla tastiera, e si comincia a lavorare nella finestra dedicata (un esempio è nella figura sotto in cui si utilizza un programma per creare immagini).

Conclusioni

L'impiego di tecniche prese da campi diversi, tra i quali la A.I., permette di incrementare in modo significativo la facilità d'uso e di comprensione dei calcolatori, riducendo inoltre il tempo necessario per l'apprendimento del loro uso. La chiave di volta può essere individuata nell'utilizzo delle immagini come mezzo di comunicazione delle informazioni. Si può vedere come queste, con le tecniche di presentazione associate, siano ormai utilizzate anche per calcolatori a larga diffusione, e si propongano quindi come lo strumento da applicare per il futuro. In un prossimo capitolo vedremo quali sono le tendenze più attuali verso cui si rivolge la ricerca per quanto riguarda le macchine indicate a supportare ambienti di sviluppo molto evoluti.

Disegno ottenuto con il programma MacPaint del Macintosh.





A sinistra, gli strumenti da adoperare (penne, matite ecc.). In basso, a sinistra gli spessori di tratto disponibili, sotto gli sfondi. La scelta viene fatta dal cursore (la freccia), pilotata dal mouse.

Lezione 20

Strutture di controllo guidate da eventi

Abbiamo visto fin qui strutture di controllo per l'iterazione caratterizzate dall'avere un unico punto di ingresso e un unico punto di uscita. Abbiamo visto in qualche caso però che l'uscita poteva essere determinata da più condizioni, che dovevano presentarsi tutte insieme (legate da AND) o in alternativa (legate da OR).
Un esempio tipico è quello della ricerca in una tabella: in tal caso ci troviamo di fronte a un ciclo di scansione della tabella stessa che viene interrotto quando l'elemento cercato viene trovato o quando gli elementi su cui effettuare la ricerca sono esauriti.
In questi casi però siamo costretti a ricorrere a variabili booleane del tipo TROVATO, che ci consentono, una volta usciti dal ciclo, di sapere su quale condizione è stato interrotto e conseguentemente di scegliere la corretta via d'esecuzione. Così per esempio il caso di una ricerca sequenziale è stato risolto nel modo seguente:

```
• Chiede elemento da cercare
• TROVATO: = FALSE
• I: = 1
• REPEAT
• IF elemento cercato = elemento di tabella (I) THEN
                • TROVATO: = TRUE
ELSE
                • I: = I + 1
• UNTIL I > numero elementi-tabella OR TROVATO
• IF TROVATO THEN
                • Visualizza "elemento presente"
ELSE
                • Visualizza "elemento assente"
```

Come si vede,all'uscita dall'iterazione va controllato il valore della variabile TROVATO per sapere che esito ha avuto la ricerca. Poiché tali situazioni sono frequenti nella programmazione, è stata ideata una struttura di controllo che consente di individuare eventi specifici e di trattarli nell'ambito di un segmento di programma. Con l'uso di tale struttura l'algoritmo di ricerca precedente si presenta così:

```
• I: = 1
• EXECUTE UNTIL TROVATO, NONTROVATO
      • EVENT TROVATO IF elemento di tabella (I) = elemento cercato
      • I: = I + 1
      • EVENT NONTROVATO IF I > numeroelementi-tabella
• THENCASE
      • WHEN TROVATO
                • Visualizza "elemento presente"
      • WHEN NONTROVATO
                • Visualizza "elemento assente"
• ENDEXECUTE
```

La struttura sopra riportata presenta un segmento iniziale che contiene l'iterazione della scansione di tabella e una porzione successiva che contiene il trattamento specifico dei due casi dichiarati inizialmente, cioè il caso di "elemento trovato" e il caso di "elemento non trovato". Le due porzioni sono separate dalla parola chiave THENCASE. Il segmento iniziale compreso tra EXECUTE e THENCASE è iterato fino a quando si verifica uno degli eventi previsti. La parola chiave EVENT specifica che, se la condizione indicata è vera, siamo nel caso dell'evento indicato.
L'iterazione viene pertanto interrotta e il controllo passa alla porzione di codice introdotta dalla corrispondente WHEN. Quando tale porzione di codice è stata eseguita la struttura termina e il controllo passa all'istruzione immediatamente seguente, la ENDEXECUTE. In tal modo abbiamo ottenuto l'obiettivo di interrompere l'iterazione sulla base, in questo caso, di due differenti condizioni e di introdurre il trattamento specifico per ciascuna senza bisogno di ricorrere a una variabile booleana. I vantaggi, in particolare in termini di semplicità e di leggibilità, ottenuti con questa struttura sono tanto più evidenti quanto più numerose sono le condizioni o meglio gli eventi che si possono presentare e che richiedono un trattamento specifico. Ciascuno di essi infatti con l'uso di una struttura classica richiederebbe una variabile booleana ad hoc con un'evidente crescita di complessità del programma e una conseguente diminuzione di leggibilità.
La struttura sopra presentata è detta, dal nome del suo inventore, struttura di Zahn. Oltre a situazioni come quella su presentata, la struttura di Zahn è particolarmente adatta per il trattamento di problemi diagnostici, in tutti quei casi cioè in cui sia richiesta una successione di controlli, ciascuno dei quali può rilevare una situazione anomala che richiede di essere identificata. Infatti uno dei problemi più complessi da elaborare con algoritmi è il cosiddetto "trattamento delle condizioni eccezionali" (exception handling). Talvolta un problema di per sé semplice e chiaro può divenire complesso qualora si voglia tenere conto di tutte le condizioni d'errore che possono capitare: in tal caso, infatti, il modello di soluzione iniziale può venire "complicato" da tutti i controlli e i relativi trattamenti. La possibilità di avere una struttura che:
— evidenzi il modello base scelto per l'algoritmo;
— individui i tipi di "eccezioni" esaminati (i nomi degli eventi dopo EXECUTE);
— individui con chiarezza i punti di rilevazione di tali condizioni d'errore;
— individui con chiarezza i punti di trattamento delle stesse (WHEN)
rende più semplice e più leggibile il programma.
Può essere usata anche una variante di tale struttura che non prevede l'iterazione iniziale. Così per esempio il controllo di correttezza di una data fornita dall'utente può essere realizzato nel modo seguente:

```
• Chiede data nella forma gg, mm, aa
• EXECUTE UNLESS GIORNOERRATO, MESEERRATO, ANNOERRATO, DATACORRETTA
      • EVENT GIORNO ERRATO IF GG<1 OR GG>31
      • EVENT MESEERRATO IF MM <1 OR MM>12
      • EVENT ANNOERRATO IF AA<0 OR AA>85
      • EVENT DATACORRETTA
• THENCASE
      • WHEN GIORNOERRATO
            • Visualizza "errore di giorno"
```

L'algoritmo esegue in successione i controlli di correttezza a meno del verificarsi di una condizione d'errore. In tal caso passa al trattamento corrispondente. Se nessuno dei controlli effettuati produce errore, tutta la porzione di codice dopo la THENCASE viene ignorata e l'esecuzione procede con l'istruzione seguente la ENDEXECUTE. In questo caso pertanto il segmento iniziale viene eseguito una sola volta.

- WHEN MESEERRATO
  - Visualizza "errore di mese"
- WHEN ANNOERRATO
  - Visualizza "errore di anno"
- WHEN DATACORRETTA
  - Elaboradata
- ENDEXECUTE

L'algoritmo così scritto corrisponde al seguente realizzato con IF:

- Chiede data nella forma gg,mm,aa
- IF GG<1 OR G>31 THEN
  - Visualizza "errore di giorno"

ELSE
- IF M<1 OR M>12 THEN
  - Visualizza "errore di mese"

ELSE
- IF AA<0 OR AA>85 THEN
  - Visualizza "errore di anno"

ELSE
- Elaboradata

che è evidentemente molto meno leggibile del precedente.

Si noti che la struttura di Zahn, sia nella forma iterativa che nell'altra, ammette tanti eventi quanti sono richlesti dal problema, ma evidenzia il verificarsi di uno solo tra questi: in tal caso infatti il trattamento normale è abbandonato per la gestione dell'evento. Gli eventi sono pertanto mutuamente esclusivi. Si noti inoltre come l'ultimo evento non sia determinato da una condizione ma soltanto dal fatto che l'esecuzione ha preso una determinata strada. Se pertanto nessuna delle condizioni d'errore si è presentata siamo evidentemente nel caso di DATACORRETTA senza bisogno di verificare ulteriori condizioni.

Si consideri anche l'esempio seguente:

- Chiede elemento da cercare
- EXECUTE UNLESS TROVATO, NONTROVATO
  - FOR I: = 1 TO numero-elementi DO
    - EVENT TROVATO IF elemento cercato = elemento di tabella (I)
  - EVENT NONTROVATO
- THENCASE
  - WHEN TROVATO
    - Visualizza "elemento presente"
  - WHEN NONTROVATO
    - Visualizza "elemento assente"
- ENDEXECUTE

Questo esempio corrisponde al primo esempio con l'uso dell'istruzione FOR; se l'intera tabella viene scandita senza che si verifichi l'evento trovato, ci troviamo evidentemente nella condizione di evento non trovato. Introduciamo quindi direttamente l'evento NONTROVATO senza bisogno di specificare condizioni.

La struttura di Zahn in BASIC

Il BASIC non dispone della struttura di Zahn: possiamo però come nel caso di altre strutture di controllo, "simularla" attraverso le istruzioni che il linguaggio ci mette a disposizione, aggiungendo i commenti che aiutino il lettore a capire le nostre intenzioni. Così l'esempio della ricerca tabellare può essere così risolto:

```
10 INPUT "Elemento richiesto";E
20 LET I=1
30 ' Execute until trovato,nontrovato
```

```
40 ' Event trovato if elemento cercato=elemento
scandito
50 IF E=T(I) GOTO 150
60 LET I=I+1
70 ' Event nontrovato if I>numero elementi
tabella
80 IF I>N GOTO 200
90 GOTO 30
100 ' Thencase
150 ' When trovato
160 PRINT "Elemento presente"
170 GOTO 250
200 ' When nontrovato
210 PRINT "elemento assente"
250 ' Endexecute
```

Si osservi come la struttura di Zahn altro non sia che un "rivestimento sintattico" espressivo di semplici istruzioni quali IF e GOTO.

Il caso del controllo della data può essere realizzato così:

```
10 INPUT "data (gg,mm,aa)";G,M,A
20 ' Execute unless giornoerrato,meseerrato,an
noerrato,datacorretta
30 ' Event giornoerrato if G<1 or G>31
40 IF G<1 OR G>31 GOTO 150
50 ' Event meseerrato if M<1 or M>12
60 IF M<1 OR M>12 GOTO 200
70 ' Event annoerrato if A<0 or A>85
80 IF A<0 OR A>85 GOTO 250
83 ' Event datacorretta
85 GOTO 270
90 ' Thencase
150 ' When giornoerrato
160 PRINT "Errore di giorno"
170 GOTO 300
200 ' When meseerrato
210 PRINT "Errore di mese"
220 GOTO 300
250 ' When annoerrato
260 PRINT "Errore di anno"
265 GOTO 300
270 ' When datacorretta
280 ' Elaboradata
300 'Endexecute
```

Come si vede, oltre a facilitare la lettura, l'uso della struttura di Zahn è un ausilio alla costruzione del programma stesso secondo uno schema di corretta impostazione, in quanto tale meno esposto all'errore o comunque più adatto a guidare il programmatore nella ricerca di eventuali errori.

Cosa abbiamo imparato

In questa lezione abbiamo appreso:

- le strutture di controllo guidate da eventi
- la realizzazione in BASIC della struttura di Zahn in forma iterativa e selettiva

INFORMATICA E MATERIE UMANISTICHE

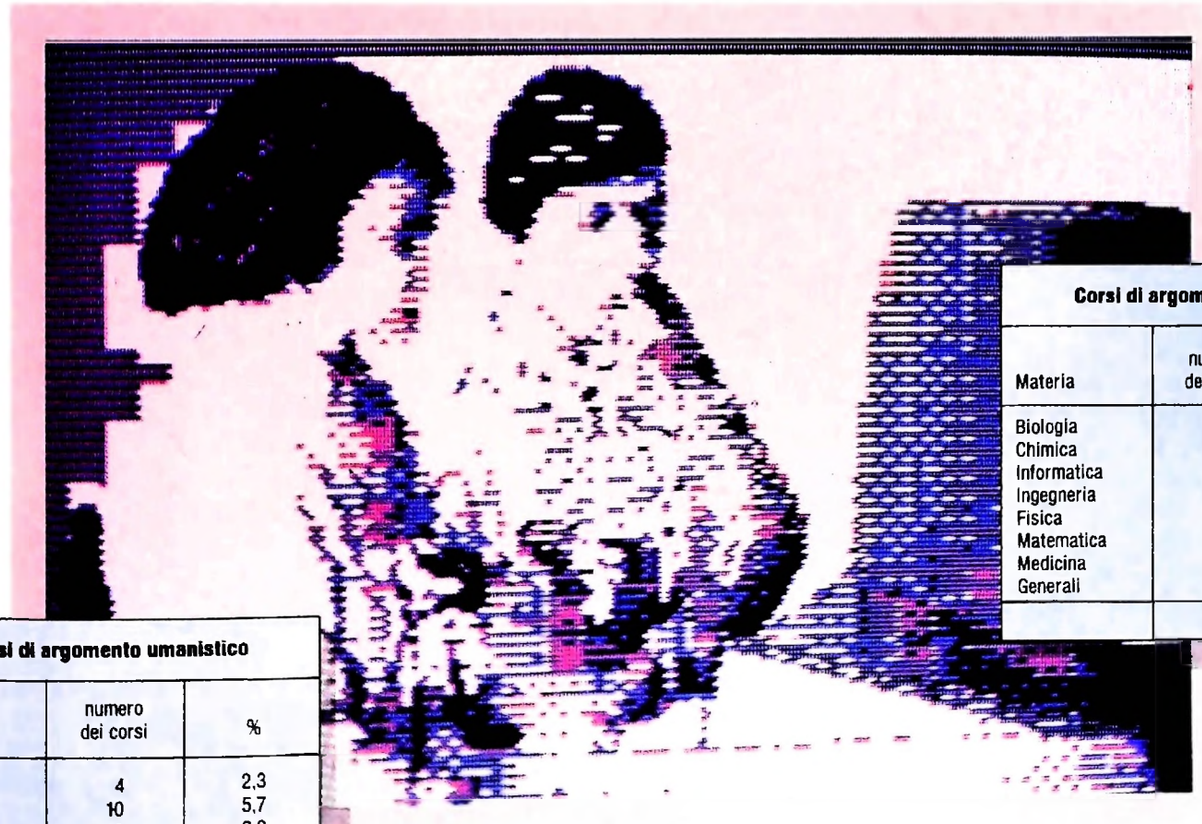
Un nuovo metodo didattico che permette un apprendimento più approfondito e realistico.

Quando si parla di Istruzione Assistita da Calcolatore (C.A.I.) e si presenta la situazione italiana, normalmente si tende ad affermare che il C.A.I. è uscito dall'ambito della sperimentazione e si sottolineano le possibilità di impiego nella didattica di discipline scientifiche. Per quanto riguarda invece le scienze umane, ci si limita a rilevare un certo interesse, in particolar modo per l'insegnamento delle lingue straniere moderne.

Queste considerazioni sono certamente corrette, ma la diffusione del C.A.I. in Italia è senza dubbio minore di quella riscontrabile in molti altri paesi, e per nessuna materia si può per ora parlare di un impiego generalizzato dei calcolatori nella didattica.

Ma, se è vero che le cose stanno per cambiare anche nel nostro paese, non è difficile prevedere che l'impiego del C.A.I. figurerà ai primi posti anche nelle materie umanistiche, come è già avvenuto, per esempio, in Francia, in Inghilterra e negli Stati Uniti.

Infatti, anche se una qualsiasi rassegna non può certamente illustrare tutte le attività esistenti, dall'esame delle tabelle



Corsi di argomento umanistico

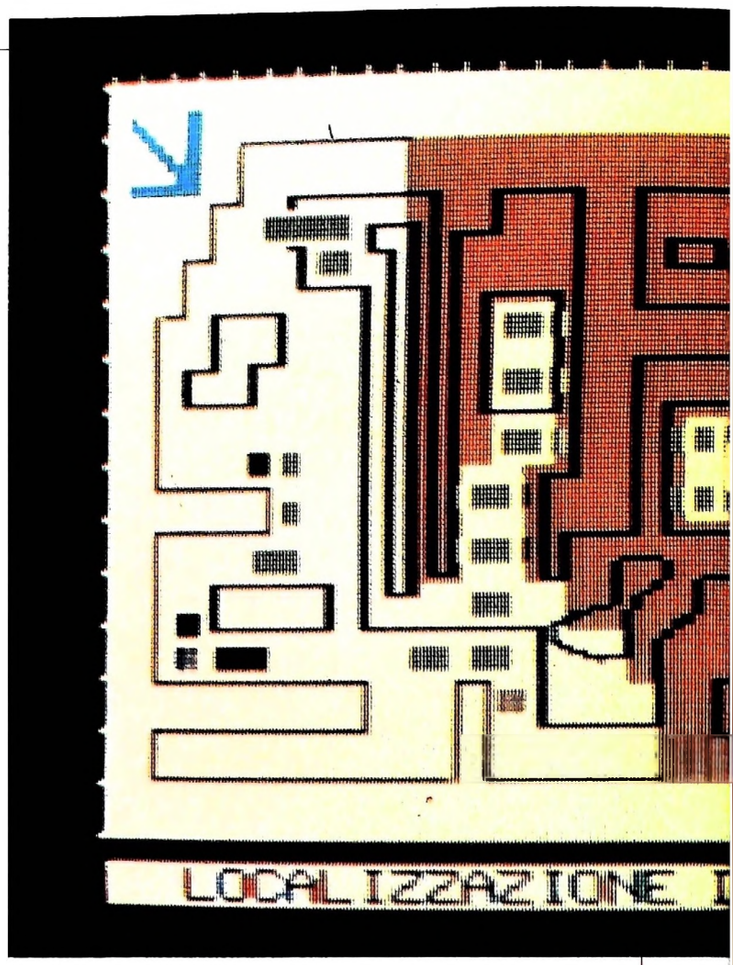
Materia	numero dei corsi	%
Arte	4	2,3
Geografia	10	5,7
Lingue	16	9,2
Linguistica	8	4,6
Musica	10	5,7
Generali	18	10,3
	66	37,8

Corsi di argomento scientifico

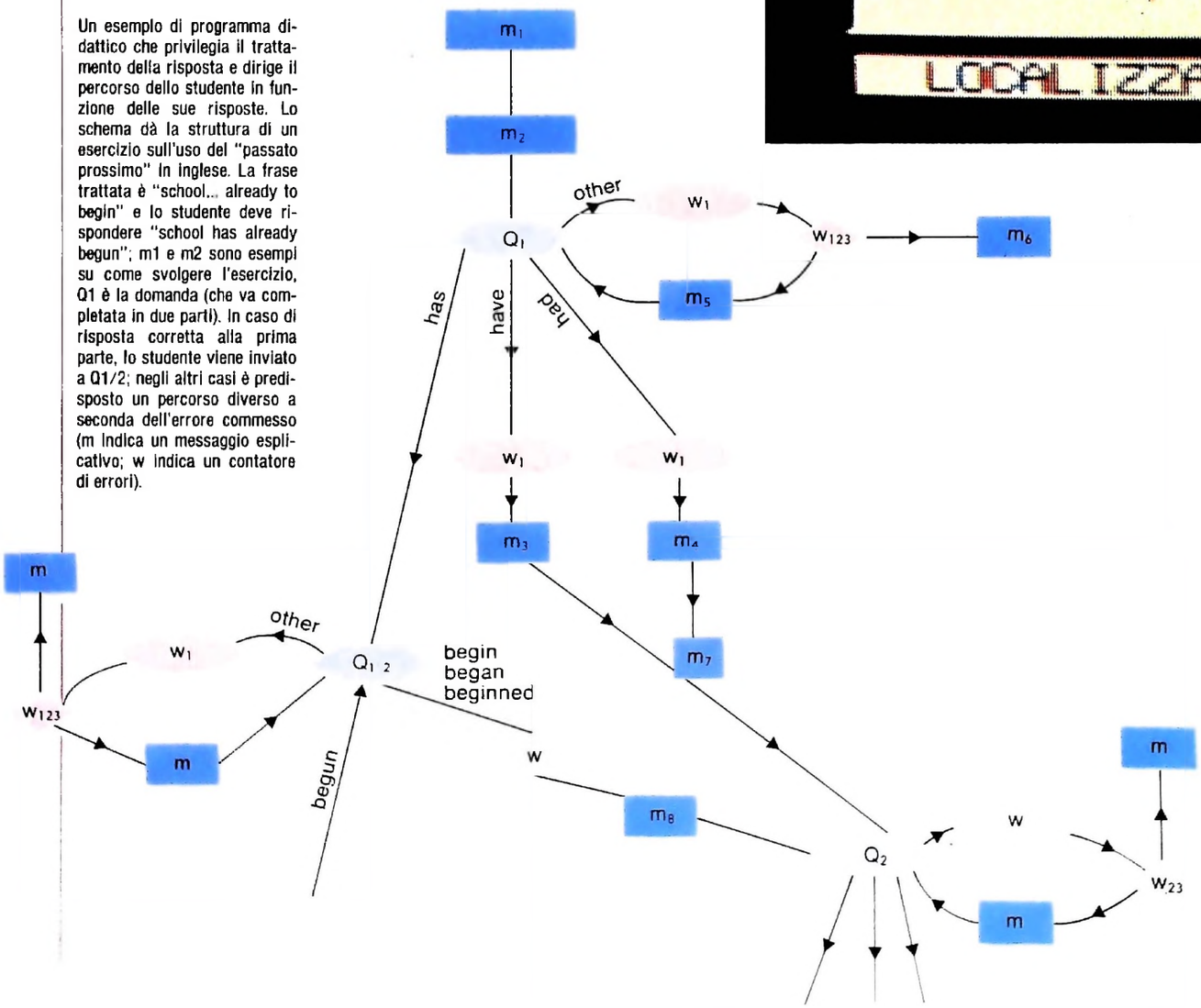
Materia	numero dei corsi	%
Biologia	6	3,4
Chimica	14	8,04
Informatica	12	6,9
Ingegneria	10	5,7
Fisica	12	6,9
Matematica	26	14,9
Medicina	14	8,04
Generali	14	8,04
	108	61,92

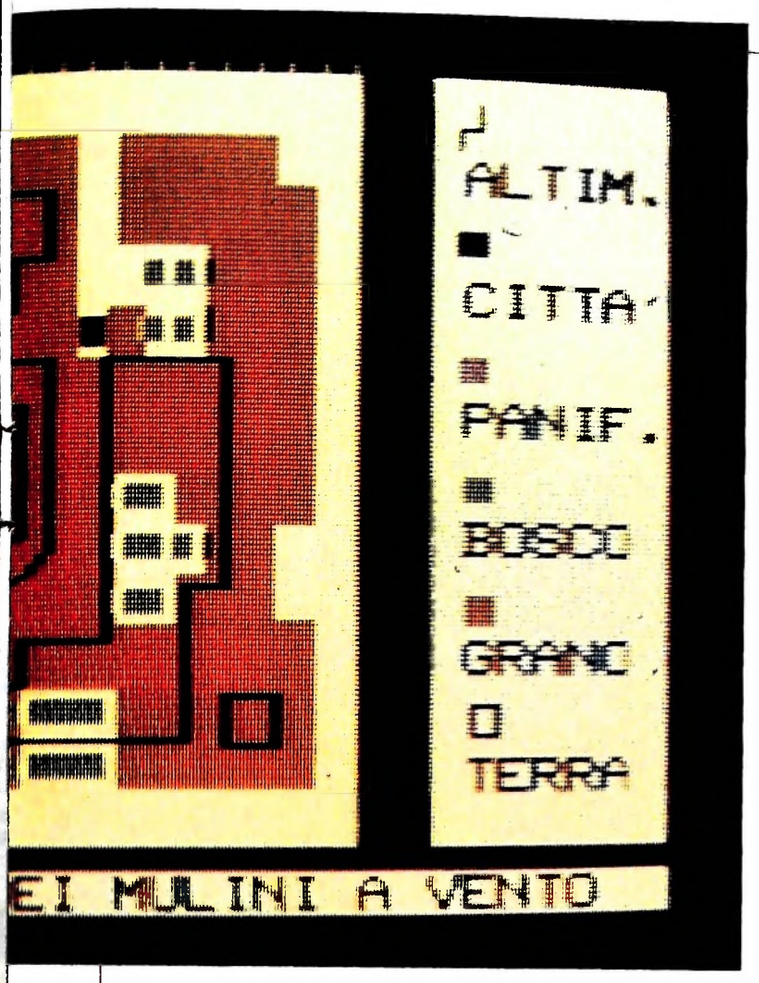
Circa il 40% dei corsi di istruzione assistita dal calcolatore (C.A.I.) riguardano materie umanistiche: le tabelle riportate sono state compilate sulla base di varie fonti italiane ed estere

qui in basso si nota come il 40% circa dei corsi C.A.I. riguardi argomenti quali le lingue straniere, l'arte e la geografia.
Questo dato dimostra come anche nel settore umanistico vi sia una grande richiesta di metodi didattici che consentano di operare con studenti che hanno una formazione di base assai differente, e come sia in atto un notevole sforzo per rendere l'insegnamento il più rispondente possibile alle effettive necessità.
Prima di esporre in dettaglio i possibili impieghi del C.A.I. in alcune materie (lingue straniere, archeologia, geografia), va sottolineato come l'uso dei calcolatori in questo caso sia assai significativo anche da un punto di vista generale e metodologico.
Infatti l'impiego del C.A.I. nell'ambito delle scienze umane si rivela particolarmente utile in quanto il più delle volte, obbliga a ridefinire l'intera materia del corso in chiave algoritmica: è quindi evidente che in questo modo l'informatica può già essere introdotta nella fase di trattamento del materiale di partenza ed è fuori dubbio che ciò rappresenta un aspetto formativo di rilevante importanza da un punto di vista generale e metodologico.



Un esempio di programma didattico che privilegia il trattamento della risposta e dirige il percorso dello studente in funzione delle sue risposte. Lo schema dà la struttura di un esercizio sull'uso del "passato prossimo" in inglese. La frase trattata è "school... already to begin" e lo studente deve rispondere "school has already begun"; m1 e m2 sono esempi su come svolgere l'esercizio, Q1 è la domanda (che va completata in due parti). In caso di risposta corretta alla prima parte, lo studente viene inviato a Q1/2; negli altri casi è predisposto un percorso diverso a seconda dell'errore commesso (m indica un messaggio esplicativo; w indica un contatore di errori).





Rielaborazione al calcolatore della mappa dell'isola usata nel "Windmill Location Game", un gioco di simulazione realizzato in Inghilterra per lo studio della geografia. Sulla base della configurazione dell'isola e della direzione dei venti, lo studente deve posizionare nel luogo più opportuno sei mulini a vento.

Il C.A.I. nell'insegnamento delle lingue

L'insistenza dell'insegnamento tradizionale sulla grammatica, l'importanza data alle esercitazioni nei metodi audio-orali e alla ripetizione in quelli audio-visivi derivano dal fatto che non si hanno a disposizione metodi efficaci per guidare l'apprendimento.
Gran parte degli insuccessi, registrabili tra l'altro anche nell'insegnamento della lingua materna, derivano dall'aver ridotto gran parte dell'apprendimento linguistico a semplice addestramento.
Nell'insegnamento mediante tecniche C.A.I. è invece possibile far "scoprire" i modelli linguistici e farli costruire dall'alunno stesso, e la costruzione dei modelli avviene spontaneamente perché l'alunno trova i mezzi per ristrutturare il proprio bagaglio sintattico e grammaticale provando di volta in volta in situazioni nuove.
Infatti in questi ultimi anni le tecniche C.A.I. nel settore linguistico hanno subito una notevole evoluzione.
La maggior differenza rispetto ai primi programmi C.A.I. consiste nel fatto che allora il calcolatore veniva prevalentemente impiegato come macchina per "far correggere" e non come macchina "per correggere".
Se il C.A.I. consente di trattare le risposte di tutto un gruppo individualmente, simultaneamente, obiettivamente e rigorosamente, vi possono essere due modi diversi e spesso complementari di concepire un programma didattico.
Si può infatti decidere di privilegiare il lavoro ripetitivo moltiplicando gli esercizi che consentiranno di apprendere la nozione. Il programma si svolgerà in modo rigido, indicando all'alunno l'errore e, su richiesta, la risposta corretta, le spiegazioni grammaticali e il vocabolario che si riferiscono all'esercizio, attribuendo eventualmente un punteggio cumulativo. Il programma è scritto con lo scopo di richiamare nozioni già affrontate e che devono essere o precisate o rinforzate o valutate. Il programma segnala l'errore, gli attribuisce un coefficiente, permette la consultazione di un dizionario o di una grammatica limitata al contesto dell'esercizio. I commenti sono brevi e lo svolgimento del programma principale dipenderà anche dalle richieste dello studente che guida così l'esecuzione del modulo di programma a partire dalle indicazioni ricevute. Gran parte dei corsi sino ad ora realizzati per l'insegnamento delle lingue sono di questo tipo, sia quelli prodotti negli Stati Uniti (particolarmente attive sono le università di Washington, la Stanford University, l'Università del Missouri e altre) sia quelli patrocinati dall'Istituto Nazionale per la Ricerca Pedagogica francese. Su questa linea si situano pure i prodotti inglesi ed italiani.
Il secondo metodo che si può prendere in considerazione consiste nel privilegiare il trattamento della risposta e nel dirigere il percorso dell'alunno in funzione delle risposte stesse. Il procedimento è allora ad albero e prevede non soltanto risposte corrette, ma anche errate o parzialmente esatte. A ogni risposta è quindi associato un commento esplicativo: questo secondo tipo di programma è stato sinora poco sviluppato perché è molto lento da mettere in opera e suppone una buona padronanza sia pedagogica sia del mezzo informatico e, in genere, può essere preso in considerazione solo nella misura in cui certi aiuti alla programmazione sono forniti da un linguaggio autore. A titolo di esempio riportiamo la descrizione di un esercizio completo di risposte (l'esempio è tratto da un programma per M20 - linguaggio Olimaster) relativo all'uso del "near past": la frase trattata è "school..... already (to begin)" e lo studente deve rispondere "school has already begun" (diagramma nella pagina a fronte).
La tendenza di fondo che emerge in modo evidente è comunque quella di produrre, in tutti i settori, moduli didattici per rendere più agevoli eventuali cambiamenti e per facilitare possibili accostamenti interdisciplinari. Ogni modulo esaurisce un particolare argomento e l'allievo può tralasciare argomenti noti per accedere direttamente a quelli che maggiormente lo interessano o che intende ripetere. In questo modo i momenti dell'apprendimento e dell'applicazione a problemi nuovi di quanto appreso vengono separati: nell'insegnamento delle lingue l'elaboratore non viene quasi mai utilizzato nella prima fase e tende ad essere usato, in collegamento con altre periferiche, prevalentemente nella seconda.

La fotografia a destra illustra un'esercitazione di "archeologia al calcolatore" relativa al ritrovamento di alcuni manufatti. Sullo sfondo elaborazione al calcolatore di graffiti di Altamira eseguita da Adriano Abbado.

ADRIANO ABBADO

Il C.A.I. e lo studio della geografia

Come è noto un calcolatore può essere impiegato per la simulazione di sistemi dinamici a variazione continua o discreta. Infatti un processo che nella realtà si svolge in tempi anche assai lunghi, può essere simulato su un elaboratore in tempi brevi e con buoni risultati. È quindi del tutto scontato che la simulazione tramite calcolatore può avere un notevole interesse didattico, ad esempio per l'economia, la geografia, e in genere per tutte quelle discipline in cui interessa illustrare le conseguenze derivanti da scelte di carattere organizzativo ed economico.
Di particolare interesse sono i corsi di geografia realizzati in Inghilterra all'interno del progetto "Computers in the curriculum", rivolto a studenti delle scuole secondarie superiori: tale sperimentazione ha interessato, a partire dal 1974, tutte le materie e il C.A.I. è stato impiegato in svariati modi ("drill and practice", "problem-solving", simulazione e giochi ecc.). I risultati conseguiti sono molto positivi ed il materiale prodotto è stato spesso preso a modello da molti: a titolo di esempio nella pagina precedente si riporta un gioco-simulazione rivolto a studenti di 12-14 anni. Il calcolatore fornisce all'allievo alcuni parametri relativi alla configurazione di un'isola e la direzione dei venti e lo studente deve posizionare nel luogo più opportuno sei mulini a vento. Il gioco prescinde ovviamente da considerazioni di politica economica, che saranno se mai svolte dal docente, e tende a rinforzare le capacità di analisi su una carta geografica, a muoversi sulle coordinate, a leggere le rappresentazioni in scala ecc.

C.A.I. ed archeologia

Per lo studente di archeologia è ovviamente di fondamentale importanza l'apprendimento delle tecniche di scavo. In passato queste sono state insegnate solo attraverso corsi teorici e campi estivi, ma la teoria non fornisce la necessaria esperienza e questo può provocare, durante lo scavo, la perdita di dati importanti.
Recentemente, per ovviare a queste difficoltà si sono creati luoghi di scavo artificiali: esiste però un metodo diverso che consente di fornire agli studenti un minimo di competenza, e cioè la simulazione con il calcolatore.
L'informatica è stata sino ad ora usata, ad esempio, per fornire campionature e analisi tecniche in un corso teorico e per fornire dati su un sito reale in un corso di tipo storico.
Lo "scavo simulato" offre due vantaggi:
1) lo studente regola la propria velocità di apprendimento;
2) i risultati di ogni quadrettatura scavata sono forniti immediatamente allo studente che può così decidere subito come proseguire.
Durante lo scavo il calcolatore fornisce una serie di informazioni e di grafici aggiuntivi sul sito archeologico, qualora lo richieda l'istruttore o lo studente. Inoltre la simulazione di uno scavo, dal momento che si usano dati di un luogo realmente scavato, può essere usata per saggiare la validità di una serie di campionature mettendo a confronto ogni scavo simulato con i risultati dello scavo reale.

Conclusioni

Come visto in precedenza, alcuni settori delle scienze umane in apparenza distanti dal C.A.I. possono invece trarre dall'impiego di tali tecniche notevoli vantaggi, permettendo agli studenti lo svolgimento di attività altrimenti impossibili.
Per quanto riguarda le materie linguistiche in particolare, abbiamo potuto verificare che il C.A.I. permette di meglio conoscere e capire il profilo linguistico degli alunni ed è non solo uno strumento di valutazione, ma anche di apprendimento assai efficace.
Il C.A.I. è particolarmente utile nei settori dove l'errore è frequente perché permette di trattarlo in modo più approfondito e diverso dai mezzi didattici tradizionali.

BASIC E FUNZIONI MUSICALI

Come usare alcune strutture di controllo del linguaggio BASIC per la descrizione di strutture musicali.

Esaminiamo in queste pagine le strutture musicali fondamentali del nostro discorso, cioè la sequenza, l'iterazione, l'alternativa e il non determinismo, e il modo in cui è possibile descriverle con il BASIC.

Le strutture di base

Il costrutto più semplice è la *sequenza* che, in un testo musicale, corrisponde alla sequenza nel tempo di eventi musicali; la trascrizione di un brano musicale mediante una successione di istruzioni SOUND corrisponde ad una sequenza di note sul pentagramma.
Possiamo descrivere una sequenza come una subroutine costituita da un blocco di istruzioni come il seguente:

```
xx00 REM sequenza
xx10 SOUND y1,z1
xx20 SOUND y2,z2
.....
.....
xxj0  SOUND yn,zm
xxk0 RETURN
```

Quando abbiamo una situazione di *iterazione* di una sequenza o, più generalmente, dell'esecuzione di una certa sequenza di istruzioni, usiamo un costrutto del tipo seguente:

```
xx10 REM iterazione
xx20 FOR i=n TO m
xx30 REM f(i)
xx40 .....
xxj0 NEXT i
```

L'iterazione ci serve anche per effettuare una particolare funzione ripetuta: la *scansione* di una tabella; nel costrutto precedente abbiamo, in questo caso, una f(A(i)) dove A(i) è l'i-esima componente del vettore A.
La scansione di tabelle (anche a più dimensioni) è tipica delle applicazioni musicali sia quando *leggiamo* sia quando *scriviamo* un testo musicale, o anche quando *valutiamo* una funzione dipendente da una certa sequenza musicale.

Quando vogliamo descrivere il verificarsi di una certa situazione in dipendenza dalla valutazione di una certa *condizione*, usiamo il costrutto IF ... THEN ...; inoltre, se in alternativa vogliamo descrivere il verificarsi di un'altra situazione, usiamo il costrutto più generale IF ... THEN ... ELSE ...
La struttura tipo è la seguente:

```
xx10 IF condizione
        THEN GOSUB yy10
        ELSE GOSUB zz10
```

Se vogliamo lasciare al *caso* il verificarsi o meno di una certa condizione, possiamo usare la generazione pseudocasuale mediante l'istruzione RND, che genera numeri reali compresi tra 0 e 1.
Un costrutto di questo tipo ha la struttura seguente:

```
xx10 REM struttura non deterministica
xx20 rn=RND(x)
xx30 IF condizione-1(rn) THEN GOSUB aa10
xx40 IF condizione-2(rn) THEN GOSUB bb10
.....
```

Combinazione di strutture di controllo

Le strutture di controllo possono essere combinate tra loro; possiamo quindi porre un'alternativa all'interno di un'iterazione oppure eseguire un'iterazione in alternativa ad un'altra; di particolare interesse sotto questo profilo è la combinazione di iterazioni *innestate*.
Un costrutto di questo genere ha la seguente struttura:

```
xx10 REM iterazioni innestate
xx20 FOR i1=n1 TO m1
xx30 FOR i2=n2 TO m2
xx40 FOR i3=n3 TO m3
xx50 REM f(i1,i2,i3)
xx60 .....
xxj0  NEXT i3
xxk0 NEXT i2
xx10 NEXT i1
```

In questo modo, la funzione f viene applicata variando l'indice i3 per ogni valore di i2 a sua volta variato per ogni valore di i1; i1 varia quindi più lentamente, i3 più velocemente.

Questo costrutto ci può servire per la scansione di una tabella tridimensionale o per l'iterazione per un certo numero di volte (m1 — n1 + 1) della scansione di una tabella bidimensionale (di dimensioni (m2 — n2 + 1) * (m3 — n3 + 1)) o per altro ancora.

Un programma per applicare le funzioni musicali

A conclusione del discorso sulle funzioni musicali nell'elaborazione automatica di testi musicali discutiamo ora il semplice programma riportato in queste pagine. È un programma che ci permette di elaborare una sequenza di note mediante le funzioni musicali di seguito elencate:

— accelerazione del tempo (subroutine 3000-3199)

— rallentamento del tempo (subroutine 3200-3299)

— trasposizione delle altezze (subroutine 4000-4199)

— retrogradazione delle altezze (subroutine 4200-4399)

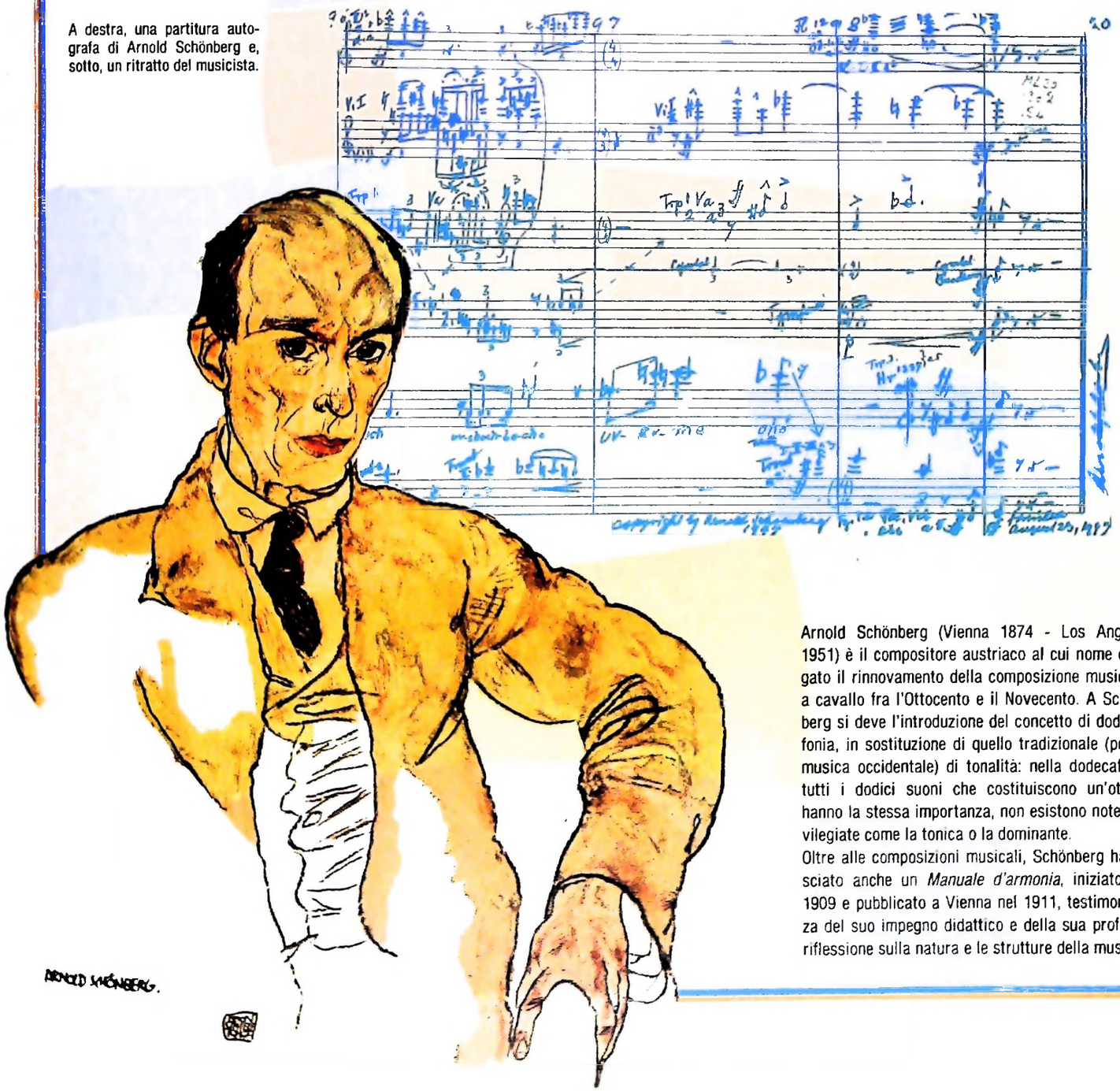
— inversione speculare delle altezze (subroutine 4400-4599).

```
010 REM programma per l'esecuzione
    di funzioni musicali
015 print "inizio della sessione"
020 clear
025 defint a,d
027 defdbl b
030 dim alt(11),dur(11),aa(11)
    ,dd(11),bb(11)
040 REM caricamento del tema
    iniziale
042 input "di quante note e' il
    tema";ln
050 for i=0 to ln-1
060 print "carica la nota";i+1;
    "del tema";
065 input alt(i),dur(i)
070 next i
080 print "la sequenza di
    altezze e':"
090 for i=0 to ln-1
100 print alt(i);
110 next i
115 print ""
120 print "la sequenza di
    durate e':"
130 for i=0 to ln-1
140 print dur(i);
150 next i
160 REM copia del tema originale
    sul buffer
165 rp$=""
170 for i=0 to ln-1
180 aa(i)=alt(i)
190 dd(i)=dur(i)
200 next i
205 gosub 9999
210 REM scelta della funzione
215 print ""
220 cn$=""
225 input "modifica altezze o
    durate (A,D)";ad$
230 if ad$="D" then gosub 1000
240 if ad$="A" then gosub 2000
245 if ad$<>"A" and ad$<>"D"
    goto 225
250 ad$=""
260 input "vuoi ascoltare";as$
270 if as$="si" then gosub 9999
280 as$=""
290 input "vuoi continuare a
    modificare";cn$
300 if cn$="si" goto 220
310 input "vuoi ripartire
    dall'inizio";rp$
320 if rp$="si" goto 80
990 print "fine della sessione"
999 end
1000 REM funzioni sulle durate
1010 print "funzioni sulle durate"
1020 input "scelta funzione
     (A,R)";fd$
1030 if fd$="A" then gosub 3000
1040 if fd$="R" then gosub 3200
1045 if fd$<>"A" and fd$<>"R" goto
     1020
1050 print "nuova sequenza di
     durate:"
1060 for i=0 to ln-1
1070 print bd(i);
1080 next i
1090 print ""
1900 fd$=""
1999 return
2000 REM funzioni sulle altezze
2010 print "funzioni sulle altezze"
2020 input "scelta funzione
```

```
      (T,R,I)";fa$
2030  if fa$="T" then gosub 4000
2040  if fa$="R" then gosub 4200
2050  if fa$="I" then gosub 4400
2055  if fa$<>"T" and fa$<>"R" and
      fa$<>"I" goto 2020
2060  print "nuova sequenza di
      altezze:"
2070  for i=0 to 1n-1
2080  print aa(i);
2090  next i
2100  print ""
2900  fa$=""
2999  return
3000  REM accelerazione
3010  input "fattore di
      accelerazione";ft
3020  for i=0 to 11
3030  bb(i)=dd(i)/ft
3035  dd(i)=bb(i)
```

Arnold Schönberg

A destra, una partitura autografa di Arnold Schönberg e, sotto, un ritratto del musicista.



Arnold Schönberg (Vienna 1874 - Los Angeles 1951) è il compositore austriaco al cui nome è legato il rinnovamento della composizione musicale, a cavallo fra l'Ottocento e il Novecento. A Schönberg si deve l'introduzione del concetto di dodecafonia, in sostituzione di quello tradizionale (per la musica occidentale) di tonalità: nella dodecafonia tutti i dodici suoni che costituiscono un'ottava hanno la stessa importanza, non esistono note privilegiate come la tonica o la dominante.

Oltre alle composizioni musicali, Schönberg ha lasciato anche un *Manuale d'armonia*, iniziato nel 1909 e pubblicato a Vienna nel 1911, testimonianza del suo impegno didattico e della sua profonda riflessione sulla natura e le strutture della musica.

```
3040 next i
3050 ft=0
3199 return
3200 REM rallentamento
3210 input "fattore di
     rallentamento";ft
3220 for i=0 to ln-1
3230 bb(i)=dd(i)*ft
3240 if bb(i)>255 then bb(i)=255
3245 dd(i)=bb(i)
3250 next i
3260 ft=0
3299 return
4000 REM trasposizione
4010 input "quanti semitoni";tn%
4020 for i=0 to ln-1
4030 bb(i)=aa(i)/1.059463^tn%
4040 if bb(i)>16383 then bb(i)
     =16383
4050 if bb(i)<311 then bb(i)=311
4055 aa(i)=bb(i)
4060 next i
4070 tn%=0
4199 return
4200 REM retrogradazione
4210 for i=0 to ln-1
4220 bb(i)=aa(ln-1-i)
4230 next i
4240 for i=0 to ln-1
4250 aa(i)=bb(i)
4260 next i
4399 return
4400 REM inversione speculare
4410 input "rispetto a quale
     altezza";tn%
4420 for i=0 to ln-1
4430 bb(i)=tn%^2/aa(i)
4440 if bb(i)<311 then bb(i)=311
4450 if bb(i)>16383 then bb(i)
     =16383
4455 aa(i)=bb(i)
4460 next i
4470 tn%=0
4599 return
9999 REM esecuzione del tema
     modificato
10000 for i=0 to ln-1
10010 sound aa(i),dd(i)
10020 next i
10030 return
```

In questo programma, il testo musicale si carica nella forma prevista dall'istruzione SOUND: altezza, durata. Il testo viene caricato nei vettori alt e dur, rispettivamente per i parametri di altezza e di durata; vengono poi usati altri tre vettori per le operazioni previste dalle funzioni musicali: aa è un vettore di interi che contiene l'ultima versione ottenuta mediante l'applicazione di funzioni della sequenza di altezze; dd svolge un ruolo analogo per le durate; bb è invece un vettore di reali in doppia precisione che serve per le operazioni aritmetiche previste dalle funzioni (sia per le altezze che per le durate).

Dopo il caricamento del tema musicale visualizziamo le sequenze delle altezze e delle durate e ascoltiamo l'esecuzione del tema (080-205); dopo ogni operazione possiamo scegliere di ascoltare il nuovo tema modificato (260-280); possiamo poi scegliere se continuare a modificare (290-300) o anche se ripartire dal tema iniziale (310-320).

Il dialogo per l'applicazione delle funzioni (210-250) si articola in:

— scelta se operare su altezze (A) o durate (D);

— se operazione sulle durate (1000-1999): scelta tra funzioni di accelerazione (A) o rallentamento (R);

— se operazione sulle altezze (2000-2999): scelta tra funzioni di trasposizione (T), retrogradazione (R), inversione speculare (I).

Dopo ogni operazione sulle durate abbiamo la stampa della nuova sequenza di durate e analogamente per le altezze.

Usiamo la routine 9999-10030 per l'esecuzione del tema musicale, sia quando vogliamo l'esecuzione del tema originale sia quando vogliamo l'esecuzione del tema ottenuto mediante l'ultima modificazione.

Per ripristinare il tema originale usiamo la sequenza 160-200 che copia il tema originale dai vettori alt e dur ai vettori buffer aa e dd.

In questo modo, dopo aver caricato un tema musicale possiamo trasformarlo mediante l'applicazione di tante funzioni quante vogliamo, con la facoltà di vedere i nuovi temi ottenuti e di ascoltarne l'esecuzione; se non siamo soddisfatti possiamo riprendere dall'inizio senza dover ricaricare il tema originale. Ma attenzione:

— ricordiamo che se applichiamo due volte la stessa funzione di *inversione speculare* o di *retrogradazione* otteniamo la sequenza di partenza; cioè, I(I(alt)) = alt e R(R(alt)) = alt;

— se produciamo valori di altezze o di durate che vanno al di fuori dei subrange accettabili dall'istruzione SOUND, il programma appiattisce questi valori ai margini (superiori o inferiori, a seconda: 3240, 4040-4050, 4440-4450).

Ricordiamo, infine, che se chiediamo una funzione di trasposizione dobbiamo indicare un numero intero corrispondente al numero di semitoni di cui vogliamo trasporre il tema (numero positivo per la trasposizione verso l'acuto e negativo verso il grave: vedi 4010 e 4030); per l'operazione di inversione speculare dobbiamo invece specificare rispetto a quale *altezza specchio* applicare la funzione: indichiamo quindi un numero intero del tipo del primo parametro dell'istruzione SOUND (vedi 4410 e 4430).

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

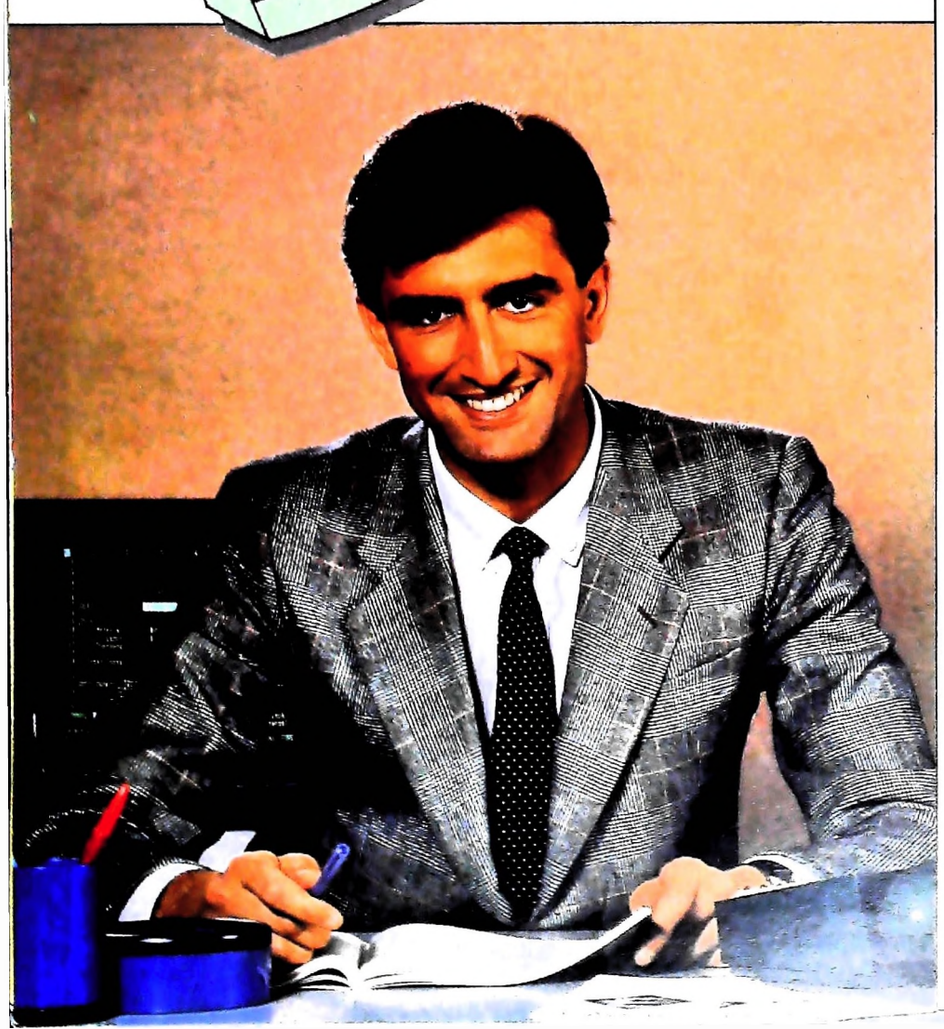
L'UFFICIO DA VIAGGIO

Anche in leasing con Olivetti Leasing.

olivetti

UN NUOVO MODO DI USARE LA BANCA.

Consulenza



GLI INVESTIMENTI CON VOI E PER VOI DEL BANCO DI ROMA.

Il Banco di Roma non si limita a custodire i vostri risparmi. Vi aiuta anche a farli meglio fruttare. Come? Mettendovi a disposizione tecnici e analisti in grado di offrirvi una consulenza di prim'ordine e di consigliarvi le forme di investimento piú giuste. Dai certificati di deposito ai titoli di stato, dalle obbligazioni alle azioni, il Banco di Roma vi propone professionalmente le varie opportunità del mercato finanziario. E grazie ai suoi "borsini", vi permette anche di seguire, su speciali video, l'andamento della Borsa minuto per minuto.

Se desiderate avvalervi di una gestione qualificata per investire sui più importanti mercati mobiliari del mondo, i fondi comuni del Banco di Roma, per titoli italiani ed esteri, vi garantiscono una ampia diversificazione.

Inoltre le nostre consociate Figeroma e Finroma forniscono consulenze per una gestione personalizzata del portafoglio e per ogni altra esigenza di carattere finanziario.

Venite ci a trovare, ci conosceremo meglio.

BANCO DI ROMA

CONOSCIAMOCI MEGLIO.