IL MIO COMPUTER

CORSO PRATICO DI INFORMATICA: MICRO, MINI E PERSONAL COMPUTER



ISTITUTO GEOGRAFICO DE AGOSTINI

IL MIO COMPUTER

CORSO PRATICO DI INFORMATICA: MICRO, MINI E PERSONAL COMPUTER

volume settimo

ISTITUTO GEOGRAFICO DE AGOSTINI

Direzione: Achille Boroli - Adolfo Boroli
Direzione editoriale: Mario Nilo; *settore fascicoli:* Jason Vella

Redazione dell'edizione italiana a cura di:
Edizioni Remo Sandron - Firenze
Redazione dell'edizione inglese:
Richard Pawson (*Editor*); David Tebbutt (*Consultant Editor*); Chris Cooper (*Executive Editor*); David Whelan (*Art Director*); Gareth Jefferson, Roger Ford (*Collaborators*)
Designed and Produced by Bunch Partworks Limited
Traduzione a cura di: Daniel Quinn

Coordinamento grafico: Otello Geddo
Coordinamento fotografico a cura del Centro Iconografico dell'Istituto Geografico De Agostini

Titolo originale dell'opera: The home computer advanced course
© Orbis Publishing Limited, Londra, 1983.
© Istituto Geografico De Agostini S.p.A., Novara, 1985.
Stampato in Italia - IGDA Officine Grafiche, Novara, 1985.

Sommario

1441 Il Music Maker
1443 Un esempio da seguire
1446 Nuotare o annegare
1449 DATABASE
1450 un computer da bambini
1452 Traguardo finale
1455 Missione di ricognizione
1458 Le chiamate OSWORD del BBC
1461 I Bulletin Board
1464 Il percorso critico
1466 Linguaggio disciplinato
1468 DATABASE
1469 Enterprise Sixty Four
1472 Interfaccia Spectrum
1474 La gestione dei file sul BBC
1477 La Foresta Incantata
1480 Ghostbusters
1481 Il computer insegna
1484 Prime parole
1486 Il Project della Microsoft
1488 DATABASE
1489 L'Apricot Portable
1492 Realizziamo l'interfaccia
1495 Ancora sui file del BBC
1497 Digitaya
1501 Scrivere col computer
1504 Tipi di variabili
1506 Un gioco di simulazione
1508 DATABASE
1509 Il CX5M della Yamaha
1512 Leggere la Bibbia
1514 Collegare l'interfaccia
1517 Il nuovo e il vecchio
1520 Elite
1521 Necessità particolari
1524 Un programmatore di EPROM
1526 Decisioni costruttive
1529 DATABASE
1530 Il Mouse della AMX
1532 Tutti a bordo!
1534 Tartarughe... scolastiche
1536 Interazioni del BASIC
1538 A me gli occhi!
1541 Viaggio nella didattica
1544 Il computer... agricolo
1546 Alimenti per la ciurma
1548 DATABASE
1549 L'Electron Plus 3
1552 Una sofisticata "mano"
1554 1+2=20?
1557 Interrompiamo il programma...
1560 Zaxxon
1561 L'allievo comanda
1564 Piccoli robot
1566 Ultimi acquisti
1569 DATABASE
1570 Hybird Music 500
1572 Suonala ancora, RAM!
1574 Braccio di ferro
1577 Organizzazione dei dati
1579 Gli eventi principali
1581 Una banca dati in classe
1584 Usare un database
1586 Pronti a salpare
1589 Il Sanyo MBC-550
1592 Montiamo il braccio
1594 Gli insiemi nel PASCAL
1597 Ultima chiamata
1600 Software in vetrina
1601 Linguaggi moderni
1604 Una "chiave" da cercare
1606 Muscoli e presa
1608 DATABASE
1609 Il Rotronics Wafadrive
1612 In alto mare
1614 A tempo di... record
1617 Il SO del Commodore 64
1621 Il Computer Assisted Learning
1624 Uomo in mare!
1626 Le matrici in PASCAL
1629 DATABASE
1630 Fine di un braccio
1632 Massima immissione
1634 Il Discovery 1
1637 I/O sul Commodore 64
1641 Una lezione di economia
1644 Ottime relazioni
1646 Linee di comunicazione
1648 DATABASE
1649 Il Penman Plotter
1652 Navigatori solitari
1655 A portata di mano
1658 Linguistica strutturale
1661 Le classi del futuro
1664 Variabili di valore
1666 Database a doppio ingresso
1668 DATABASE
1669 Il kit Fischertechnik
1672 In un mare di guai
1675 Olio di gomito
1678 Effetti ottici
1680 "Knight Lore"

Indice per settore

Hardware

1441 Il Music Maker
1450 Un computer da bambini
1469 Enterprise Sixty Four
1489 L'Apricot Portable
1509 Il CX5M della Yamaha
1524 Un programmatore di EPROM
1530 Il Mouse della AMX
1549 L'Electron Plus 3
1564 Piccoli robot
1570 Hybird Music 500
1589 Il Sanyo MBC-550
1609 Il Rotronics Wafadrive
1634 Il Discovery 1
1649 Il Penman Plotter
1669 Il kit Fischertechnik

Panorami

1443 Un esempio da seguire
1461 I Bulletin Board
1481 Il computer insegna
1501 Scrivere col computer
1521 Necessità particolari
1541 Viaggio nella didattica
1561 L'allievo comanda
1581 Una banca dati in classe
1600 Software in vetrina
1601 Linguaggi moderni
1621 Il Computer Assisted Learning
1641 Una lezione in economia
1661 Le classi del futuro

Programmazione BASIC

1446 Nuotare o annegare
1477 La Foresta Incantata
1497 Digitaya
1506 Un gioco di simulazione
1532 Tutti a bordo!
1546 Alimenti per la ciurma
1566 Ultimi acquisti
1586 Pronti a salpare
1612 In alto mare
1624 Uomo in mare!
1652 Navigatori solitari
1672 In un mare di guai

Linguaggio macchina



1449 DATABASE
1458 Le chiamate OSWORD del BBC
1468 DATABASE
1474 La gestione dei file sul BBC
1488 DATABASE
1495 Ancora sui file del BBC
1508 DATABASE
1517 Il nuovo e il vecchio
1529 DATABASE
1536 Interazioni del BASIC
1548 DATABASE
1557 Interrompiamo il programma...
1569 DATABASE
1579 Gli eventi principali
1597 Ultima chiamata
1608 DATABASE
1617 Il SO del Commodore 64
1629 DATABASE
1637 I/O sul Commodore 64
1646 Linee di comunicazione
1648 DATABASE
1668 DATABASE
1678 Effetti ottici

Note d'Informatica



1452 Traguardo finale
1466 Un linguaggio disciplinato
1484 Prime parole
1504 Tipi di variabili
1526 Decisioni costruttive
1554 1+2=20?
1572 Suonala ancora, RAM!
1594 Gli insiemi nel PASCAL
1614 A tempo di... record
1626 Le matrici in PASCAL
1658 Linguistica strutturale
1664 Variabili di valore

Laboratorio



1455 Missione di ricognizione
1472 Interfaccia Spectrum
1492 Realizziamo l'interfaccia
1514 Collegare l'interfaccia
1538 A me gli occhi!
1552 Una sofisticata "mano"
1574 Braccio di ferro
1592 Montiamo il braccio
1606 Muscoli e presa
1630 Fine di un braccio
1655 A portata di mano
1675 Olio di gomito

Software



1464 Il percorso critico
1480 Ghostbusters
1486 Il Project della Microsoft
1512 Leggere la Bibbia
1520 Elite
1534 Tartarughe... scolastiche
1544 Il computer... agricolo
1560 Zaxxon
1577 Organizzazione dei dati
1584 Usare un database
1604 Una "chiave" da cercare
1632 Massima immissione
1644 Ottime relazioni
1666 Database a doppio ingresso
1680 "Knight Lore"

Il Music Maker

Quali effetti si possono ottenere con questa curiosa combinazione hardware/software?



UN "PIANOFORTE" DA TAVOLINO

Il package "Music Maker", progettato per il Commodore 64, si compone di una piccola tastiera musicale ad incastro e di un programma di controllo. La tastiera a due ottave viene fissata alla custodia del computer e i suoi tasti azionano quelli del micro, appositamente programmati per generare le corrispondenti note musicali. La tastiera lascia libero accesso ai tasti funzione, che vengono usati per programmare i suoni. Il pacchetto nel suo complesso produce suoni di elevata qualità

IAN McKINNELL

Alcune caratteristiche proprie degli home computer hanno indotto numerose industrie elettroniche a produrre periferiche capaci di esaltarne le potenzialità. Le funzioni maggiormente interessate sono quelle grafiche e quelle audio, per le quali sono state progettate numerose apparecchiature. Molte di esse sono dedicate al Commodore 64, essendo questo privo anche dei più semplici comandi di suono, presenti invece su altri micro. L'apparecchio può contare su numerosi package musicali che, indipendentemente dalla loro qualità, permettono di generare suoni solo attraverso la tastiera del computer e non su una tastiera simile a quelle di un pianoforte. Tuttavia, con il *Music Maker*, la Commodore offre un sistema che consente di accedere direttamente al chip SID (Sound Interface Device) del CBM 64 tramite una tastiera con tasti bianchi e neri tipo pianoforte, da collocare sopra la tastiera normale.
La custodia di rivestimento in plastica rigida viene fissata ai bordi del computer e circonda i tasti alfabetici e di funzione. La tastiera musicale, estesa su due ottave, è di plastica più morbida e i tasti, dotati di appositi prolungamenti, sono fissati individualmente alla custodia. Quando si preme un tasto, il prolungamento esercita una pressione sul tasto sottostante, programmato dal software per generare una particolare nota.
Il sistema è semplice, poco costoso e funziona egregiamente. Il rivestimento rigido mantiene ben fissa la tastiera musicale e raramente accade di "perdere" note, anche durante l'esecuzione di brani veloci e complessi.

Il software del Music Maker

Il software di corredo al package è fornito su cassetta o dischetto. Il programma è pilotato da menu, essenzialmente attraverso gli 8 tasti funzione che consentono di modificare uno o più suoni prodotti dalla tastiera intervenendo sulla forma o sull'altezza dell'onda sonora. Il programma può inoltre generare accompagnamenti ritmici di bassi o di percussioni, mentre un "sequencer" (sequenziatore) permette di programmare accordi e di suonarli al momento opportuno (in "playback"). Gli accordi possono essere memorizzati (funzione SAVE) o riletti (funzione LOAD) dalla cassetta o dal dischetto.
Ciascuno degli otto canali di suono può essere programmato in modo autonomo. Il tasto F6 permette di selezionare il canale che si vuole modificare e visualizza le seguenti opzioni: "attacco", "mantenimento", "caduta" e "rilasciamento" di una nota. A queste funzioni possono essere assegnati parametri compresi tra 0 e 15.
Quindi il programma chiede all'utente se desidera attivare uno dei filtri (i filtri "tagliano" le frequenze superiori ed inferiori a quella specificata). Se la risposta è affermativa, compare un secondo menu, mediante il quale si fissa la frequenza e il livello del filtro.
Il sequencer è una componente fondamentale della musica sintetizzata ma spesso viene trascurato in molti pacchetti musicali per microcomputer. Solitamente viene impiegato per le basi ritmiche e la sua programmazione si suddivide in due fasi: innanzi tutto il programmatore batte sulla tastiera musicale le note di una particolare sequenza, quindi ne definisce la durata, ossia il ritmo di ripetizione, con il tasto di funzione F5.
Le tre basi di percussione previste dal software del *Music Maker* sono piuttosto limitative ma sufficienti, quelle dei bassi seguono lo stesso andamento ritmico. Tre opzioni consentono di abilitare e di disabilitare l'emissione dei bassi, oppure di modificarne la tonali-

MUSIC MAKER

SOFTWARE

I programmi di gestione sono forniti su cassetta o su dischetto

DOCUMENTAZIONE

Vengono forniti due manuali a corredo del Musik Maker. La guida per l'utente contiene tutte le informazioni necessarie per il corretto funzionamento, sebbene manchino spiegazioni per gli utenti più esperti. Un secondo manuale contiene utili informazioni sulla notazione musicale e tutta una serie di spartiti

PRO

Considerato il suo basso costo, il Music Maker è un ottimo affare. Consente infatti di sfruttare al massimo le capacità musicali del Commodore 64, con effetti davvero strabilianti

CONTRO

La serie delle opzioni, sebbene ampia, ha numerose limitazioni per quanto concerne l'uso abbinato degli effetti, in particolare dell'accompagnamento ritmico

tà. A seconda della base selezionata, la tonalità della linea dei bassi può essere ridotta di metà o di un'intera ottava. La velocità del ritmo può essere modificata con i tasti per il cursore: quello di destra rallenta il tempo, quello di sinistra lo accelera.
Purtroppo molte delle opzioni previste non possono essere usate contemporaneamente. Ad esempio, il programmatore può abbinare i bassi alle percussioni ma, in tal caso, può suonare un motivo sulla tastiera solo se quest'ultima si trova nel modo monofonico: è impossibile suonare accordi in presenza di una base ritmica di sottofondo.
Tale regola vale anche per l'opzione "sequencer". Molti complessi musicali hanno rinunciato alle comuni sezioni ritmiche di bassi e percussioni sostituendole, nei loro concerti, con un sequencer. Purtroppo il *Music Maker* manca di questa funzione né può abbinare il sequencer alla sezione ritmica selezionata. Questo limite è dovuto alla struttura dell'hardware. Poiché il chip SID possiede solo tre "voci" o canali, non è possibile emettere suoni polifonici assieme ai ritmi dei bassi e delle percussioni. Anche il "glissando", ossia l'opzione che permette di elevare la tonalità di una nota premendo contemporaneamente il suo tasto e la barra spaziatrice, non funziona insieme alla sezione ritmica. Probabilmente ciò è dovuto al fatto che il suono viene generato in modo digitale, quindi il processore non è in grado di suonare i bassi e le percussioni e, contemporaneamente, modificare l'altezza del suono per cambiare la tonalità.
Più difficile da capire è il motivo per cui il software del pacchetto non consenta all'utente di programmare la macchina con nuove basi ritmiche di bassi e percussioni. Una sezione ritmica programmabile accrescerebbe la versatilità del pacchetto e le tecniche necessarie sarebbero identiche a quelle usate per il sequencer. Altre difficoltà sorgono quando un brano viene suonato in tempo reale, in quanto è impossibile modificare i parametri per il canale e l'ottava quando il sistema si trova nel modo "play": l'utente è costretto ad usare due sole ottave.
Il *Music Maker* è corredato da due manuali: la guida per l'utente contiene una serie di istruzioni per il caricamento e una succinta descrizione del funzionamento delle diverse funzioni. Pur non essendo molto dettagliata, è sufficiente per imparare a destreggiarsi tra i vari menu e per utilizzare al meglio le potenzialità del pacchetto.
Il secondo manuale, intitolato *Start Playing Keyboard*, contiene una serie di istruzioni sull'uso della tastiera e una spiegazione sulla notazione musicale. Il resto del manuale contiene le partiture di 28 canzoni famose.
Il *Music Maker* tenta di sfruttare al meglio le funzioni di suono del Commodore 64. Anche chi non si è mai interessato di musica per computer impara rapidamente ad usare il pacchetto e, dopo un breve "apprendistato" sulla tastiera, può facilmente suonare accordi e canzoni. Gli utenti più esperti possono trovare il software poco versatile e la tastiera troppo compatta per l'esecuzione dei brani più sofisticati, tuttavia il *Music Maker* resta un ottimo investimento per esplorare gli orizzonti della musica computerizzata.

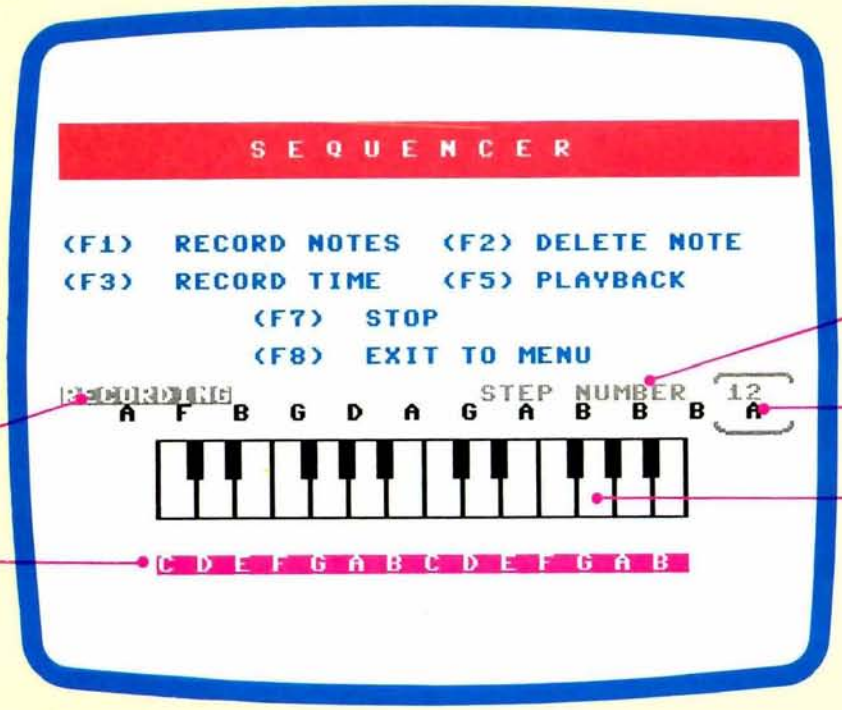
MENU MUSICALE

L'immagine riproduce il menu visualizzato dall'opzione "sequencer" del Music Maker

Il menu mostra le sei scelte possibili. Si noti che le note da suonare e i loro parametri di durata (TIME) devono essere definiti separatamente

Questo messaggio comunica all'utente che il sistema si trova nel modo "registrazione" e non in "playback" (riproduzione)

Questa linea ricorda all'utente il nome delle note presenti sulla tastiera



Un cursore blu indica all'utente la nota suonata in un particolare momento

Questo valore rappresenta il numero delle note già definite

Questo messaggio indica le note attualmente inserite nella memoria del sequencer

Un esempio da seguire?

I diversi sistemi britannici di comunicazione fra computer

I sistemi computerizzati di pubblico accesso si dividono in due categorie: i sistemi gratuiti, per i quali l'utente non deve pagare alcuna tariffa, e quelli commerciali. I tre principali sistemi commerciali in Gran Bretagna sono il Telecom Gold, il Compunet ed il Prestel (che comprende il Micronet 800). Per quest'ultimo, già ampiamente descritto in precedenza (vedere alle pagg. 581-583), vengono fornite soltanto alcune sintetiche notizie. I sistemi di tipo gratuito, più noti come "bulletin board", sono di solito gestiti da appassionati di computer attraverso comuni micro e si rivolgono essenzialmente ad altri appassionati. Di questi sistemi viene trattato in seguito. Vediamo invece le caratteristiche dei sistemi commerciali. Il Telecom Gold è il sistema di posta elettronica della British Telecom. Dedicato principalmente agli utenti commerciali, offre agli abbonati un sistema di scambio immediato di posta elettronica, oltre ad alcune funzioni telex e di archiviazione dei file. Ogni "casella postale" viene venduta ad un prezzo molto ragionevole e, dopo il primo mese di uso gratuito, all'utente vengono addebitati il tempo di accesso al sistema e l'area di immagazzinamento occupata. Esiste una tariffa minima mensile ed una oraria (più alta durante le ore d'ufficio, ridotta ad un terzo negli altri periodi della giornata). Le spese di immagazzinamento sono dovute alla posta depositata e non letta ed all'archiviazione dei file di lavoro e della posta.

Poiché il Gold è stato progettato tenendo presenti le esigenze delle grosse compagnie commerciali, la "password" di accesso al sistema è composta da un codice di tre lettere per l'identificazione della compagnia e da un codice di tre numeri che indica un particolare utente di tale compagnia.

Molte case produttrici di computer hanno acquistato numerose caselle in blocco, a prezzo scontato, rivendendole poi singolarmente ai propri clienti.

Ad esempio in Gran Bretagna, chi possiede un computer Tandy, può acquistare una casella TCC (Tandy Computer Corporation) per posta elettronica ad un prezzo molto inferiore rispetto alla normale quota d'installazione. Pur non contemplando il primo mese di uso gratuito, questa formula d'acquisto è quasi sempre vantaggiosa per l'utente. Anche il Prestel è un sistema commerciale gestito dalla British Telecom. Sebbene possieda funzioni di posta elettronica simili a quelle del Gold, il suo scopo principale è la trasmissione delle informazioni. Il Prestel è una gigantesca banca dati che offre informazioni su svariati argomenti quali i prezzi dei titoli azionari, la cucina, il lavoro, la "hit-parade" del software di maggior successo, le associazioni di appassionati di computer, le previsioni del tempo ecc.

Il Prestel si basa su una struttura ad albero: per prima cosa il sistema mostra all'utente un menu principale attraverso il quale selezionare un'area generica, ad esempio l'informatica. Quindi, passando attraverso una serie di sotto-menu e di selezioni successive, l'utente raggiunge l'argomento desiderato. Il "sentiero" percorso potrebbe essere il seguente: Micro BBC, software, software gratuito, giochi ed infine *Pacman*. Tuttavia, se l'utente conosce il numero di pagina dell'argomento che gli interessa, può saltare tutti i menu e visualizzare subito l'informazione.

La sezione del database dedicata agli appassionati di computer è stata chiamata Micronet 880. Essa consente all'utente di giocare con altri abbonati al sistema, di caricare dal database software gratuito o a pagamento, di chiedere informazioni di carattere tecnico e ricevere risposte, di leggere riviste e notizie dedicate ai microcomputer.

Per collegarsi al Micronet è necessario possedere un modem e un apposito software di comunicazione che gli utenti del Micro BBC, dello Spectrum e del Commodore 64 possono acquistare direttamente dal Micronet 800 a



KEVIN JONES

MA LE POSTE INGLESI ...

Per spedire una lettera da Londra a Glasgow via treno, col servizio Red Star occorrono 7 sterline e la lettera giunge al destinatario dopo circa 6 ore e mezzo

La stessa lettera viene recapitata dopo non meno di 24 ore se viene imbucata in una comune cassetta. Il costo è di 17 pence

La trasmissione via telex è immediata, ma occorre possedere un telex il cui impianto e canone di affitto sono piuttosto costosi. Inoltre va corrisposto un importo in base alla durata della "conversazione"

La stessa lettera, inviata con il servizio di posta elettronica del Gold, viene a costare 10,5 pence se la trasmissione avviene in un'ora di punta e solo 3,5 pence a tariffa ridotta. A questa cifra va aggiunto il costo di una chiamata telefonica della durata di un minuto

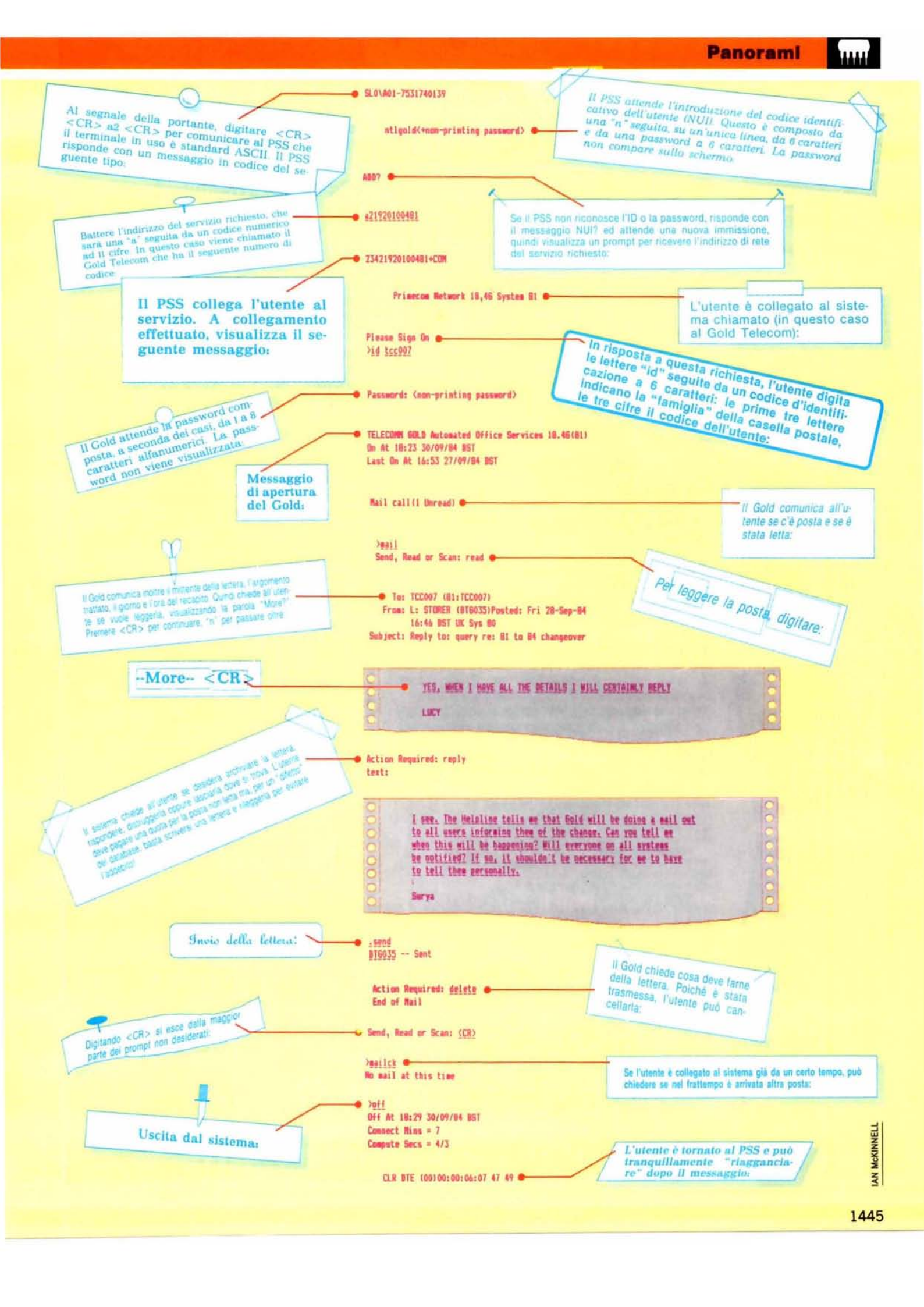
prezzi diversificati. La tariffa d'abbonamento trimestrale varia da poche sterline per l'utente privato a cifre molto più elevate per le aziende o per le scuole. Inoltre viene addebitato il tempo di accesso e il prezzo del software acquistato tramite il sistema.
Il Compunet fornisce una serie di servizi simili a quelli della sezione Micronet 800 del Prestel. È quindi possibile inviare posta elettronica, "affiggere" inserzioni, ricevere software dal database centrale e ottenere informazioni e notizie sui microcomputer.
Per collegarsi al Compunet è necessario acquistare un modem Commodore, dotato di una ROM che contiene un numero seriale distintivo e il software di comunicazione per entrare nel sistema. Quando l'utente effettua la chiamata d'accesso, il sistema lo invita a comunicare il suo numero d'identificazione, che viene confrontato con il numero seriale residente in ROM: se non corrisponde, il Compunet rifiuta l'accesso al sistema.
Questa procedura presenta vantaggi e svantaggi. Il principale vantaggio è che nessuno può adoperare un codice d'identificazione non proprio, per introdursi nel sistema, perché il numero comunicato non corrisponde a quello del modem in uso. Lo svantaggio è che non è possibile collegarsi al Compunet, pur essendo abbonati, con un modem diverso dal proprio di cui non si conosce il numero d'identificazione.
Un secondo dispositivo di protezione previsto dal sistema mira a proteggere il Compunet dai "pirati" del software. Infatti, il software fornito dal database viene "marchiato" con il numero seriale del modem usato per caricarlo e può essere eseguito solo se il modem è collegato al computer.
Come il Prestel, il Compunet presenta una struttura ad albero composta da un menu principale e da una serie di sotto-menu. Tuttavia la selezione degli argomenti è più agevole che sul Prestel perché il sistema visualizza i menu al centro dello schermo e un elenco di comandi (che può scorrere) nella parte inferiore. Quest'ultimo presenta comandi del tipo: «VISIONE dell'opzione evidenziata» oppure «Torna INDIETRO di una pagina», «COPIA la pagina corrente su nastro o disco», e così via. I tasti per i movimenti verticali del cursore vengono usati per selezionare le opzioni del menu Compunet, quelli per i movimenti orizzontali per selezionare i comandi dell'elenco.
L'abbonamento annuale al Compunet ha un costo molto modesto e i primi 12 mesi sono gratuiti. Un'ora di accesso costa circa quanto due dischetti, ma il servizio è gratuito durante la notte e nei fine settimana.
Contrariamente all'opinione corrente, il Compunet non è un servizio viewdata gestito dalla Commodore, bensì dalla compagnia indipendente Compunet Teleservice Ltd. Attualmente si possono collegare al sistema solo Commodore 64, ma è previsto un ampliamento ai più popolari micro.

Il "Packet Switching service"

Per concludere, esaminiamo il "Packet Switching Service", ancora della British Telecom, che non è un servizio per computer simile a quello del Gold, del Prestel o del Micronet, ma rappresenta un'economica via di accesso ad altri sistemi. Se l'utente non si trova nell'area di chiamata locale del sistema con cui vuole collegarsi, il costo della chiamata telefonica è notevolmente più alto.
Il PSS della British Telecom, che offre un'adeguata risposta a questo problema, è una "rete" collegata a tutti i principali sistemi di pubblico accesso del mondo. Gli utenti possono chiamare via telefono la locale "stazione" del PSS e, per suo tramite, accedere a qualsiasi sistema collegato. L'utente paga solo il costo della chiamata locale ed una tariffa fissa per l'uso del PSS. Alla pagina seguente è illustrata la procedura di accesso al Gold tramite il sistema PSS.

SISTEMI A CONFRONTO

Sistema	TELECOM GOLD	PRESTEL	COMPUNET	PSS
N. Telefonico (Dati)	01—278 4355 oppure via PSS	A seconda dell'area 300/300 baud = 01—248 5747	A seconda dell'area	A seconda dell'area
N. Telefonico (informazioni)	01—403 6777	01—278 3143	0536—205252	100 (chiedere servizio PSS)
Operatività	24 ore	24 ore	24 ore	24 ore
Baud	300/300 e 1200/1200	1200/75 (&300/300 in Londra)	1200/75	300/300 e 1200/1200
Parità	Nessuna	Viewdata	Viewdata	Nessuna
Bit Dati	8	Viewdata	Viewdata	8
Bit di stop	1	Viewdata	Viewdata	1
Soccorso	INFO INFO	*0£ (*36£ per lasciare mess.)	Selez. HELP da menu	Non previsto
Uscita	OFF	Aggancio linea telefonica	Selez. LEAVE da menu	Aggancio linea telef.
Posta elettr.?	Si	Si	Si	No
Annunci?	Si	Si	Si	No
Software a pagamento?	No	Si	Si	No
Software gratuito?	No	Si	Si	No



IAN McKINNELL



Nuotare o annegare

Ecco gli ultimi due "luoghi speciali" della nostra avventura: la palude e il villaggio











La "palude" ed il "villaggio" sono le ultime due locazioni speciali del gioco *Foresta Incantata*. Esaminiamo per prima la "palude". Come per le altre locazioni speciali, dobbiamo decidere la sua struttura prima di passare alla programmazione: lo sviluppo può essere più o meno complesso, ma si rammenti che la progettazione di una locazione complessa richiede un lavoro di programmazione più lungo e occupa una quantità di memoria maggiore.

La palude

La trama della locazione "palude" è la seguente:

Quando il giocatore entra nella palude comincia ad affondare.
Sebbene possa ricorrere a tutti i comandi normali, il giocatore riesce a salvarsi solo nuotando.
Per poter nuotare, il giocatore deve avere con sé meno di due oggetti, altrimenti il loro peso lo trascina a fondo; così, se il giocatore possiede due oggetti, deve lasciarne uno.
Tutti gli oggetti lasciati andare sono perduti per sempre e non possono essere recuperati.

```
4870 REM **** S/R PALUDE ****
4875 SF=1
4880 SN$="COMINCI AD AFFONDARE IN UNA PALUDE.": GOSUB 5500
4885 PRINT: INPUT "COMANDI";IS$
4890 GOSUB 2500: REM SUDDIVIDE IL COMANDO
4895 IF F=0 THEN 4885: REM ISTRUZIONE NON VALIDA
4900 GOSUB 3000: REM COMANDO NORMALE
4910 IF VB$="GUARDA" THEN GOSUB 2000: GOTO 4885
4915 IF VB$="LASCIA" THEN IV$(F,2)="-2": REM OGGETTO
     PERDUTO PER SEMPRE
4917 IF VF=1 THEN 4885: REM COMANDO NORMALE
4920 REM **** NUOVI COMANDI ****
4925 IF VB$<>"NUOTA" THEN SN$="NON CAPISCO":
     GOSUB 5500: GOTO 4885
4930 REM **** NUOTATA ****
4932 F=0
4935 FOR I=1 TO 2
4940 IF IC$(I) <>"" THEN F=F+1
4950 NEXT I
4955 IF F<2 THEN GOSUB 5035: RETURN: REM CONTINUA A NUOTARE
4960 GOSUB 5000: RETURN: REM POSSIEDE DUE OGGETTI
5000 REM **** S/R POSSIEDE DUE OGGETTI ****
5010 SN$="GLI OGGETTI TI APPESANTISCONO E TI TRASCINANO A FONDO.":
     GOSUB 5500
5012 PRINT: INPUT "COMANDI";IS$
5015 GOSUB 2500: REM SUDDIVIDE IL COMANDO
5020 IF VB$<>"LASCIA" THEN GOSUB 5080: REM AFFONDA
5025 GOSUB 3900: IV$(F,2)="-2": REM LASCIA UN OGGETTO
5030 IF HF=0 OR F=0 THEN 5080: REM AFFONDA
5035 REM **** CONTINUA A NUOTARE ****
5040 SN$="ADESSO PUOI ATTRAVERSARE A NUOTO LA PALUDE. QUALE
     DIREZIONE SCEGLI ?": GOSUB 5500
5050 EX$(2)="00080605": GOSUB 2300: REM DEFINISCE E VISUALIZZA LE USCITE
5055 PRINT: INPUT "COMANDI";IS$
5060 GOSUB 2500: REM SUDDIVIDE IL COMANDO
5062 IF F=0 THEN 5055: REM ISTRUZIONE NON VALIDA
5065 GOSUB 3500: REM MUOVE
5067 EX$(2)="00000000": REM NESSUNA USCITA
5070 RETURN
5075 :
5080 REM **** S/R AFFONDA ****
5085 SN$="AFFONDI NELLA PALUDE E ANNEGHI": GOSUB 5500
5090 END
```

Le linee 4870-4917 gestiscono i comandi normali per mezzo delle subroutine sviluppate in precedenza. Se il giocatore decide di abbandonare un oggetto al momento di entrare nella palude, questo viene preso in carico dalla routine LASCIA, che reinserisce la sua posizione nell'inventario principale IV$(,) usando il valore corrente del contatore di locazione P. Quindi, per il programma, l'oggetto adesso si trova nella locazione palude; se invece il giocatore abbandona l'oggetto all'interno della palude, per cancellare ogni sua traccia è necessario cambiare il corrispondente valore in IV$(,).
Si rammenti che IV$(F,2) contiene la posizione dell'oggetto F, ossia il numero di locazione, oppure −1 se l'oggetto è in possesso del giocatore. Per eliminare definitivamente l'oggetto dal gioco, si deve correggere IV$(F,2) in modo che non possa essere interpretato come un valore di posizione. Nella linea 4915, a IV$(F,2) viene attribuito il valore −2, naturalmente dopo che la routine LASCIA ha assegnato ad F l'oggetto da eliminare.
Se il giocatore decide di nuotare, il programma esamina il suo inventario personale, per controllare quanti oggetti vi sono contenuti. Se possiede meno di due oggetti, il programma chiama la subroutine CONTINUA A NUOTARE, che gli consente di lasciare la palude. Se il giocatore trasporta due oggetti, ha la possibilità di abbandonarne uno: in caso contrario, annega.
La routine CONTINUA A NUOTARE consente al giocatore di scegliere la direzione in cui vuole nuotare. L'informazione per l'uscita, contenuta nelle istruzioni DATA originali, era 00000000 e indicava che non esistevano uscite dalla palude. La linea 5050 modifica tale informazione e chiama la subroutine che descrive le uscite possibili. A questo punto il giocatore può scegliere un'uscita e allontanarsi dalla palude. Poiché l'informazione per l'uscita viene azzerata alla fine della routine, al giocatore che rientra nella palude non è consentita alcuna via d'uscita.

Il villaggio

Il villaggio costituisce la via d'uscita dalla foresta incantata, ma il giocatore deve compiere numerose operazioni prima di poter

terminare il gioco. La locazione villaggio ha la seguente struttura:

Il villaggio è protetto da un alto muro, apparentemente invalicabile.
L'unica porta d'accesso è sorvegliata da una guardia.
Per poter entrare nel villaggio, il giocatore deve prima uccidere la guardia con un colpo di fucile.
Quindi deve aprire la porta con la chiave.

La subroutine VILLAGGIO è divisa in due parti: per prima cosa occorre eliminare la guardia, poi aprire la porta. Non è difficile immaginare la scena: il giocatore, armato di fucile, si avvicina al villaggio; dopo aver ucciso la guardia però, si accorge di aver bisogno della chiave per aprire la porta. Se non la possiede, deve abbandonare la locazione villaggio per andare a cercarla. Se, al suo ritorno, sullo schermo compare la precedente descrizione della locazione villaggio, cioè con la guardia ancora in vita, il programma deve essere modificato perché questo messaggio può comparire una sola volta: se la guardia è stata uccisa, la sua presenza deve essere cancellata dalla descrizione del villaggio.
Questo problema è stato risolto usando la variabile GF come un flag che indica se la guardia è viva o morta. Inizialmente la variabile contiene il valore 0 per segnalare che la guardia è viva: se il giocatore la uccide, GF viene posta a 1. Quando il giocatore entra nella locazione del villaggio, il contenuto di GF viene esaminato per determinare se il pericolo "guardia" esiste o meno. Uccisa la guardia, il giocatore si avvicina alla porta. Le istruzioni vengono suddivise e interpretate dalle stesse routine usate in precedenza. Il giocatore conclude vittoriosamente l'avventura se possiede la chiave e la usa per aprire la porta.

```
5100 REM **** S/R VILLAGGIO ****
5102 SF=1
5105 SN$="IL VILLAGGIO È PROTETTO DA UN ALTO MURO.": GOSUB 5500
5106 IF GF<>0 THEN GOSUB 5190: RETURN: REM PORTA
5107 SN$="UNA GUARDIA SORVEGLIA L'UNICA PORTA D'ACCESSO":
     GOSUB 5500
5115 PRINT: INPUT "COMANDI";IS$
5120 GOSUB 2500: IF F=0 THEN 5115: REM ISTRUZIONE NON VALIDA
5125 GOSUB 3000: REM ISTRUZIONI NORMALI
5130 IF VB$="GUARDA" THEN GOSUB 2000: REM DESCRIVE
5135 IF VB$="VAI" AND MF=1 THEN RETURN
5140 IF VF=1 THEN 5115: REM ISTRUZIONE SUCC.
5145 IF VB$<>"UCCIDI" THEN SN$="NON CAPISCO":GOSUB 5500:
     GOTO 5115
5150 REM **** UCCIDE ****
5155 SN$="COSA USI PER UCCIDERE LA GUARDIA ?": GOSUB 5500
5160 SN$="SCRIVERE IL NOME DELL'OGGETTO OPPURE <I> PER ISTRUZIONI":
     GOSUB 5500
5162 INPUT IS$:IF IS$="I" THEN 5115
5165 GOSUB 2500: REM SUDDIVIDE
5167 IF F=0 THEN 5160: REM ISTRUZIONE NON VALIDA
5170 GOSUB 5300: IF F=0 THEN SN$="NON HAI "+W$:
     GOSUB 5500: GOTO 5160
5172 OV=F: GOSUB 5450: REM OGGETTO POSSEDUTO?
5174 IF HF=0 THEN SN$="NON POSSIEDI "+IV$(F,1):GOSUB 5500:
     GOTO 5160
5175 IF F<>1 THEN SN$=IV$(F,1)+" NON TI È DI ALCUN AIUTO":
     GOSUB 5500: GOTO 5160
5180 SN$="LA GUARDIA E' STATA UCCISA": GOSUB 5500: GF=1
5185 :
5190 REM **** S/R PORTA CHIUSA ****
5195 SN$="TI AVVICINI E CERCHI DI APRIRE LA PORTA DEL VILLAGGIO"
5200 SN$=SN$+" MA LA PORTA È CHIUSA E NON SI APRE":
     GOSUB 5500
```

```
5210 GOSUB 2500: IF F=0 THEN 5205: REM COMANDO NON VALIDO
5215 GOSUB 3000: REM COMANDO NORMALE
5220 IF VB$="GUARDA" THEN GOSUB 2000: REM DESCRIVE
5225 IF VB$="VAI" AND MF=1 THEN RETURN
5230 IF VF=1 THEN 5205: REM ISTRUZIONE SUCC.
5232 IF VB$="USA" THEN 5240
5234 IF VB$="APRI" THEN SN$="COME ?": GOSUB 5500: GOTO 5205
5235 SN$="NON CAPISCO": GOSUB 5500: GOTO 5205
5240 GOSUB 5300: REM OGGETTO VALIDO
5242 OV=F: GOSUB 5450: REM OGGETTO POSSEDUTO?
5244 IF F=0 THEN SN$=W$+" NON ESISTE!": GOSUB 5500 :GOTO 5205
5246 IF HF=0 THEN SN$="NON POSSIEDI IL "+IV$(F,1): GOSUB 5500:
     GOTO 5205
5248 IF F<>3 THEN SN$=IV$(F,1)+" NON TI È DI ALCUN AIUTO": GOSUB 5500:
     GOTO 5205
     5250 REM **** AL SICURO! ****
5255 SN$="HAI APERTO LA PORTA E, INDOSSATI GLI ABITI DELLA"
5260 SN$=SN$+" GUARDIA UCCISA, ATTRAVERSI IL VILLAGGIO SENZA ESSERE
     RICONOSCIUTO."
5265 SN$=SN$+"FINALMENTE SEI IN SALVO !": GOSUB 5500
5270 END
```

Per poter chiamare le routine di queste due locazioni speciali, occorre correggere come segue la linea 2720 della subroutine che stabilisce se una locazione è speciale o no:

```
2720 ON P GOSUB 4590, 4870, 5100, 4590
```

Basic & Basic

Spectrum:

Nel listato di "Foresta Incantata", sotituire SN$ con S$, IS$ con T$, VB$ con B$, IV$(,) con V$(,), EX$( ) con X$( ) e IC$( ) con I$( ).
Sostituire le seguenti linee:

```
2720 IF P=1 THEN GOSUB 4590
2722 IF P=2 THEN GOSUB 4870
2724 IF P=3 THEN GOSUB 5100
2726 IF P=4 THEN GOSUB 4590

4997 LET A$=IC$(I): GOSUB 7000
4998 IF A$<>"" THEN LET F=F+1

5175 IF F<>1 THEN LET A$=IV$(F,1):GOSUB 7000:
5177 IF F<>1 THEN LET S$=S$+" NON TI È DI ALCUN
     AIUTO": GOSUB 5500: GOTO 5160

5246 IF F<>3 THEN LET A$=IV$(F,1):GOSUB 7000:
5248 IF F<>3 THEN LET S$=S$+" NON TI È DI ALCUN
     AIUTO": GOSUB 5500: GOTO 5205
```

Nel listato di "Digitaya", sostituire SN$ con S$; IS$ con I$, VB$ con B$, IV$(,) con V$(,) e IC$( ) con I$(,)
Sostituire le seguenti linee:

```
3990 LET S$="IL TUO": LET A$=V$(F,1): GOSUB 7000
3995 LET S$=S$+" È INUTILE, LA PORTA CHIUSA"

4520 LET V$(4,1)=STR$(INT(RND(1)*40+8))
```

Micro BBC:

Sostituire la seguente linea nel listato "Foresta Incantata":

```
100 LET GF=0
```

Sostituire la seguente linea nel listato "Digitaya":

```
4520 IV$(4,2)=RND(40)+8
```













LISTATO DIGITAYA

```
2960 REM **** S/R PORTA UTENTE ****
2970 SF=1
2980 SN$="LA FUGA E' A PORTATA DI MANO MA
     L'IMPIEGATO ALLA PRENOTAZIONE DDR"
2990 SN$=SN$+" TI SBARRA LA STRADA.  TI INFORMA
     HA L'ORDINE DI FAR PASSARE"
3000 SN$=SN$+" SOLO INPUT. TUTTAVIA EGLI
     ACCETTA"
3010 SN$=SN$+" QUALSIASI CARTA DI CREDITO."
3020 GOSUB 5880: REM FORMATTA LA STAMPA
3030 :
3040 PRINT: INPUT "COMANDI";IS$
3050 GOSUB 1700: REM ANALIZZA IL COMANDO
3060 GOSUB 1900: REM PROCEDURE NORMALI
3070 IF MF=1 THEN RETURN: REM ESCE
3080 IF VF=1 THEN 3040: REM ISTRUZIONE SUCC.
3090 IF VB$<>"CONSEGNA" THEN PRINT
     "NON CAPISCO": GOTO 3040
3100 :
3110 REM ** L'ISTRUZIONE E' 'CONSEGNA'**
3120 GOSUB 5730: REM OGGETTO VALIDO?
3130 IF F=0 THEN PRINT "NON HAI ";NN$:
     GOTO 3040: REM ISTRUZIONE SUCC.
3140 :
3150 REM ** L'OGGETTO E' UNA CARTA
     DI CREDITO? **
3160 IF F<>5 THEN PRINT "ACCETTA SOLO CARTE DI
     CREDITO":GOTO 3040
3170 :
3180 REM ** IL GIOCATORE POSSIEDE LA CARTA
     DI CREDITO? **
3190 OV=5: GOSUB 5830
3200 IF HF=0 THEN PRINT "NON HAI ";IV$(5,1):
     GOTO 3040
3210 :
3220 SN$="L'IMPIEGATO PRENDE LA CARTA E DICE
     'MOLTO BENE, SIGNORE'"
3230 GOSUB 5880: REM FORMATTA LA STAMPA
3240 SN$="OLTREPASSI LA BARRIERA ED ENTRI
     NELLA PORTA UTENTE"
3250 GOSUB 5880: REM FORMATTA LA STAMPA
3260 :
3270 REM ** IL GIOCATORE POSSIEDE
     DIGITAYA? **
3280 OV=6: GOSUB 5830
3290 IF HF=1 THEN 3380: REM MISSIONE
     COMPIUTA
3300 :
3310 REM ** INSUCCESSO **
3320 SN$="SEI RIUSCITO A SFUGGIRE
     ALLA MACCHINA"
3330 SN$=SN$+" MA LA TUA MISSIONE E'
     FALLITA PERCHE' NON HAI"
3340 SN$=SN$+" PORTATO CON TE IL MISTERIOSO
     DIGITAYA"
3350 GOSUB 5880: REM FORMATTA LA STAMPA
3360 END
3370 :
3380 REM ** MISSIONE COMPIUTA **
3390 SN$="CONGRATULAZIONI, SEI RIUSCITO
     A LIBERARE"
3400 SN$=SN$+" IL MERAVIGLIOSO DIGITAYA
     DALLA"
3410 SN$=SN$+" MORSA DELLA MACCHINA."
3420 GOSUB 5880: REM FORMATTA LA STAMPA
3430 END
3440 :
3450 REM ** S/R PORTA PER CASSETTA **
3460 SF=1
3470 SN$="SENTI UNA FORZA IRRESISTIBILE CHE
     TI TRASCINA VERSO "
3480 SN$=SN$+"UNO STATO DI AMNESIA MAGNETICA
     PERMANENTE"
3490 GOSUB 5880: REM FORMATTA LA STAMPA
3500 NS=0: REM ISTRUZIONI DI INIZIO CONTEGGIO
3510 REM ** ISTRUZIONI **
3520 NS=NS+1: IF NS>3 THEN 3770: REM ESPULSO
3530 PRINT: INPUT "COMANDI";IS$
3540 GOSUB 1700: REM ANALIZZA IL COMANDO
3550 GOSUB 1900: REM PROCEDURE NORMALI
3560 IF MF=1 THEN MF=0: PRINT "NON RIESCI A
     MUOVERTI": GOTO 3510
3570 IF VF=1 THEN 3510: REM ISTRUZIONE SUCC.
3580 IF VB$<>"USA" THEN PRINT "NON CAPISCO":
     GOTO 3510
3590 REM ** L'ISTRUZIONE E' 'USA' **
3600 GOSUB 5730: REM OGGETTO VALIDO?
3610 IF F=0 THEN PRINT "NON HAI ";NN$:
     GOTO 3510
3620 :
3630 REM ** L'OGGETTO E' UN ATTIVATORE DEL
     BUFFER? **
3640 IF F=8 THEN 3680: REM TUTTO OK
3650 SN$="IL TUO  "+IV$(F,1)+"E' INUTILE,
     L'ATTRAZIONE AUMENTA"
3660 GOSUB 5880: GOTO 3510 :REM ISTRUZIONE
     SUCC.
3670 :
3680 OV=8: GOSUB 5830: REM IL GIOCATORE
     POSSIEDE L'ATTIVATORE?
3690 IF HF=0 THEN SN$="NON POSSIEDI "
     +IV$(8,1): GOSUB 5880: GOTO 3510
3700 :
3710 REM ** IN SALVO **
3720 SN$="USI L'ATTIVATORE DEL BUFFER
     PER OPPORTI"
3730 SN$=SN$+" ALL'AMNESIA MAGNETICA.
     LA FORZA DIMINUISCE"
3740 GOSUB 5880: REM FORMATTA LA STAMPA
3750 RETURN
3760 :
3770 REM ** ESPULSO **
3780 SN$="LA FORZA DIVIENE INSOSTENIBILE
     E VIENI ESPULSO NEL 'NULLA'"
3790 SN$=SN$+"  MAGNETICO  ATTRAVERSO  LA
     PORTA PER LA CASSETTA."
3800 GOSUB 5880: REM FORMATTA LA STAMPA
3810 END
4180 REM **** S/R DISPOSITIVO TRI-STATO ****
4190 SF=1
4200 SN$="UN GRANDE CARTELLO ANNUNCIA
     'STAZIONE DI INGRESSO/USCITA' MA,
     APPENA TI AVVICINI,"
4210 SN$=SN$+" UN CONTROLLORE GRIDA
     'BIGLIETTO, PREGO'"
4220 GOSUB 5880: REM FORMATTA LA STAMPA
4230 :
4240 REM ** ISTRUZIONI **
4250 PRINT: INPUT "COMANDI";IS$
4260 GOSUB 1700: GOSUB 1900: REM ANALIZZA
4270 IF MF=1 THEN RETURN
4280 IF VF=1 THEN 4240: REM ISTRUZIONE SUCC.
4290 IF VB$<>"MOSTRA" AND VB$<>"CONSEGNA"
     THEN PRINT "NON CAPISCO": GOTO 4240
4300 REM ** L'ISRUZIONE E' 'MOSTRA' **
4310 GOSUB 5730: REM OGGETTO VALIDO?
4320 IF F=0 THEN PRINT "NON HAI ";NN$:
     GOTO 4240:REM ISTRUZIONE SUCC.
4330  :
4340 REM ** L'OGGETTO E' UN BIGLIETTO? **
4350 IF F=4 THEN 4400; REM TUTTO OK
4360 SN$="IL CONTROLLORE SCUOTE LA TESTA E
     DICE"
4370 SN$=SN$+" QUESTO "+IV$(F,1)+" NON E'
     VALIDO"
4380 GOSUB 5880: GOTO 4240: REM ISTRUZIONE
     SUCC.
4390 :
4400 OV=4: GOSUB 5830: REM IL GIOCATORE
     POSSIEDE IL BIGLIETTO
4410 IF HF=0 THEN PRINT "NON HAI IL
     BIGLIETTO": GOTO 4240
4420 :
4430 REM ** TUTTO OK **
4440 SN$="IL CONTROLLORE FORA IL BIGLIETTO E
     TI PERMETTE"
4450 SN$=SN$+" DI SUPERARE LA BARRIERA."
4460 GOSUB 5880: REM FORMATTA LA STAMPA
4470 REM ** CANCELLA IL BIGLIETTO
     DALL'INVENTARIO **
4480 F=0
4490 FOR J=1 TO 4
4500 IF IC$(J)=IV$(4,1) THEN IC$(J)="": J=4
4510 NEXT J
4520 IV$(4,2)=STR$(INT(RND(TI)*40+8)): REM
     RIDEFINISCE LA POSIZIONE DEL BIGLIETTO
4530 P=15: MF=1: RETURN
```

DATABASE

Ecco la sesta tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi

DESCRIZIONE	INDIRIZZO
INT function routine routine funzione INT	AC78
Increment MAN#1 incrementa MAN#1	A4B6
Increment the 1AC incrementa 1AC	9222
Initialisation of language inizializzazione del linguaggio	8023
Insert a line into program inserimento di una linea nel programma	BC8D
Integer addition somma di interi	9C53
Integer comparison confronto tra interi	9AA2
Integer division divisione tra interi	99BE
Integer multiplication moltiplicazione tra interi	9D41
Integer subtraction sottrazione tra interi	9CBA
Keyword action addresses (LSB) indirizzo tabella azioni (parte bassa)	836D
Keyword action addresses (MSB) indirizzo tabella azioni (parte alta)	83E0
Keyword table tabella delle parole chiave	8071
LEFT$ function routine routine funzione LEFT$	AFCC
LEN function routine routine funzione LEN	AED1
LET statement routine routine istruzione LET	8BE4
LEVEL 1 of expression evaluator calcolo espressioni LIVELLO 1	9B29
LEVEL 2 of expression evaluator calcolo espressioni LIVELLO 2	9B72
LEVEL 3 of expression evaluator calcolo espressioni LIVELLO 3	9B9C
LEVEL 4 of expression evaluator calcolo espressioni LIVELLO 4	9C42
LEVEL 5 of expression evaluator calcolo espressioni LIVELLO 5	9DD1
LEVEL 6 of expression evaluator calcolo espressioni LIVELLO 6	9E20
LIST command routine routine comando LIST	B59C
LISTO command routine routine comando LISTO	B58A
LN function routine routine funzione LN	A7FE

INDIRIZZO	DESCRIZIONE
9EDF	**Convert number to a string** conversione di un numero in stringa
9EDF	**Print number in decimal** visualizza un numero in decimale
A07B	**Convert a string to a number** conversione da stringa a numero
A178	**Set MAN#1 = MAN#1 + MAN#2** pone MAN#1 = MAN#1 + MAN#2
A197	**Set MAN#1 = MAN#1 * 10** pone MAN#1 = MAN#1 * 10
A1DA	**Ceck FAC#1** controllo FAC#1
A1F4	**Set FAC#1 = FAC#1 * 10** pone FAC#1 = FAC#1 * 10
A21E	**Set FAC#2 = FAC#1** pone FAC#2 = FAC#1
A23F	**Set FAC#2 = FAC#1 * 2** pone FAC#2 = FAC#1 * 2
A2A4	**Set MAN#1 = MAN#1 + A** pone MAN#1 = MAN# + A
A2BE	**Set FAC#1 = IAC** pone FAC#1 = IAC
A2ED	**Set FAC#1 = A** pone FAC#1 = A
A303	**Normalise FAC#1** normalizza FAC#1
A34E	**Unpack FAC#2 from (&4B)** decompatta FAC#2 da (&4B)
A37D	**Pack FAC#1 to &471 onwards** compatta FAC#1 da &471 in poi
A381	**Pack FAC#1 to &476 onwards** compatta FAC#1 da &476 in poi
A385	**Pack FAC#1 to &465 onwards** compatta FAC#1 da &465 in poi
A38D	**Pack FAC#1 to (&4B)** compatta FAC#1 da (&4B)
A3B2	**Unpack FAC#1 from (&4B)** decompatta FAC#1 da (&4B)
A3E4	**Set IAC = INT(FAC#1)** pone IAC = INT(FAC#1)
A3FE	**Fix FAC#1** fissa FAC#1
A453	**Zero FAC#2** azzera FAC#2
A486	**Set FAC#1 = FAC#1 —INT(FAC#1)** pone FAC#1 = FAC#1 —INT(FAC#1)
A4B6	**Increment MAN#1** incrementa MAN#1
A4C7	**Decrement MAN#1** decrementa MAN#1

Un computer da bambini

Un'opportunità per insegnare ai più piccoli gli elementi della lingua inglese

L'apparecchio My Talking Computer della Microspeech è dedicato ai bambini con più di tre anni ed è stato adottato da numerose scuole elementari della Gran Bretagna. La sua leggera custodia è in materiale plastico e misura 23x25 cm. Le "mascherine" grafiche per i diversi programmi vengono appoggiate su un pannello sensibile al tatto, posto sul davanti della macchina e bloccate da un telaietto mobile. Da questo punto di vista l'apparecchio ricorda la lavagna grafica Touchmaster (vedere a pag. 1310), ma le sue funzioni sono notevolmente diverse. Al My Talking Computer è abbinato un altro apparecchietto, My Talcking Clock che, usato assieme ad uno dei programmi incorporati, insegna ai bambini a "leggere" l'orologio.

Sulla sinistra del computer c'è un alloggiamento per l'innesto di cartucce ROM di programmi. L'impostazione didattica seguita dal My Talking Computer è innovativa, ma le applicazioni dei programmi interni sono piuttosto limitate e si rivolgono ad una fascia d'età abbastanza ristretta. I progettisti hanno cercato di superare questo limite dotando il sistema di un connettore d'espansione per l'introduzione di ROM aggiuntive. La prima ROM prodotta, detta Expansion Module One, è leggermente più sofisticata della ROM interna dell'apparecchio e la Microspeech sembra aver intenzione di sviluppare una serie di programmi di crescente complessità per seguire lo sviluppo del bambino. Sopra il connettore d'espansione c'è un piccolo altoparlante. L'apparecchio può essere alimentato da 5 pile da 1,5 volt oppure dalla corrente di rete.

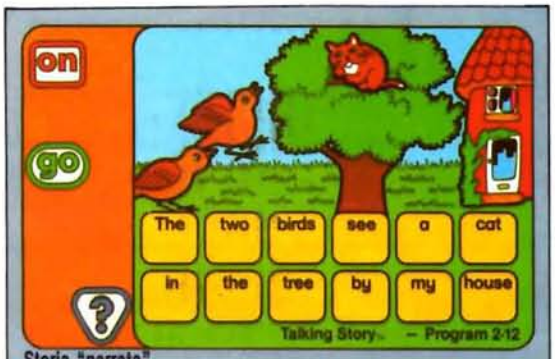
I programmi sono contenuti in un singolo chip ROM con una capacità, secondo la casa, di oltre 120 Kbyte. Il software ROM è composto da cinque programmi principali, ciascuno suddiviso in numerosi sottoprogrammi dedicati allo stesso argomento. Il primo è rivolto all'insegnamento dell'aritmetica e contiene esercizi per il riconoscimento dei numeri e delle quattro operazioni elementari. Il secondo insegna ad associare le parole alle immagini. Il terzo, detto calcolatrice parlante, pronuncia i numeri e i simboli aritmetici digitati dal bambino e comunica a voce i risultati. Il quarto programma si basa su una serie di giochi per il riconoscimento delle parole: il computer pronuncia una parola ed il bambino deve battere il tasto corrispondente. Il quinto usa il My Talking Clock per insegnare al bambino a leggere l'orologio.

Il bambino seleziona i programmi sfogliando un libro composto da una serie di mascherine, poi pone quella prescelta a contatto del pannello frontale e la fissa sull'apposito telaietto; quindi preme la casella ON per accendere la macchina e una voce femminile ri-

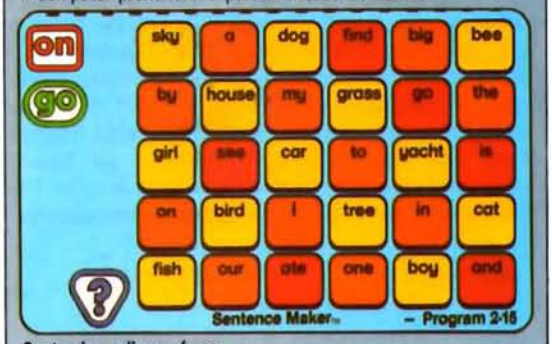


Le prime parole

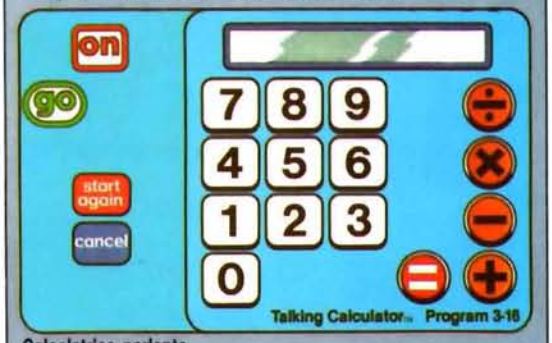
Il My Talking Computer, grazie alla semplicità del suo funzionamento, è un ottimo strumento per avvicinare i bambini al mondo dei computer. Le mascherine grafiche del "libro" dei programmi vengono appoggiate su un pannello sensibile al tatto e fissate con un telaietto. L'alloggiamento per moduli d'espansione che contengono i programmi accessori si trova sulla sinistra dell'apparecchio



Storia "narrata"
Questa mascherina è per bambini da 2 a 12 anni. Il computer riconosce il programma selezionato dalla posizione dei "pulsanti" ON e GO. Quando il bambino preme uno dei quadratini, il computer pronuncia la parola in esso contenuta



Costruzione di una frase
Il programma insegna a riconoscere le parole scritte. Quando viene premuto il punto interrogativo, il computer pronuncia una parola e il bambino deve digitarne il tasto.



Calcolatrice parlante
Forse il programma più interessante di My Talking Computer è il Talking Calculator. Il bambino preme i numeri o i simboli aritmetici e il computer calcola e comunica il risultato



IAN McKINNELL

Lettura dell'orologio
Sul pannello viene appoggiata anche la mascherina di My Talking Clock. Dal retro dell'orologio sporgono una serie di piedini che toccano il pannello rendendo l'apparecchio più preciso. Le lancette si possono ruotare sul quadrante e il computer comunica al bambino l'ora indicata

sponde pronunciando la parola «Hello» ed invita il bambino a premere il tasto GO. Il computer riconosce il programma in uso dalla diversa posizione dei tasti ON e GO sulle mascherine. Questa semplice tecnica di selezione è particolarmente indicata per i bambini più piccoli che possono caricare ed eseguire qualsiasi programma senza l'intervento di un adulto, compresa la funzione dei tasti ON e GO. Le mascherine sono rivestite da un sottile strato plastico che le protegge da eventuali danni (rovesciamento di liquidi, macchie, ecc.).
La valutazione dei diversi programmi didattici segue criteri abbastanza precisi: il pacchetto deve veramente insegnare gli argomenti per cui è stato progettato e non deve dare per scontata nessuna precedente conoscenza da parte del bambino. Ad esempio, un programma che insegna a leggere non deve contenere pagine d'istruzioni scritte e inoltre deve essere divertente e facile da usare.

Il sintetizzatore di parole

La caratteristica più interessante di My Talking Computer è il sintetizzatore vocale interno all'apparecchio, che sarebbe molto utile anche sulle comuni calcolatrici. Tuttavia anche questa funzione presenta alcuni limiti: le parole vengono memorizzate come tali e successivamente richiamate una alla volta per costruire le frasi. Quindi la formulazione delle frasi risulta piuttosto rigida e una stessa parola viene spesso usata in contesti diversi. Poiché My Talking Computer è prodotto "oltre oceano", l'accento della voce femminile è spiccatamente americano e non rispecchia i canoni del più puro stile britannico, pur essendo comunque accettabile.
Il principale difetto del sistema, soprattutto per una macchina progettata per fini didattici, è che alcune parole risultano leggermente indistinte, anche ripetendo più volte le frasi che le contengono.
Ecco una valutazione complessiva di My Talking Computer, in base ai criteri prima ricordati. La facilità di caricamento e di esecuzione dei nuovi programmi è indubbiamente positiva; altrettanto significativa per l'insegnamento della lettura ai bambini è la capacità di parlare del computer. Nonostante la poca chiarezza di alcune parole, l'abbinamento tra emissione vocale e forma scritta rappresenta un metodo d'insegnamento molto più diretto di quello adottato da altri pacchetti software che associano più semplicemente le parole alle immagini. Inoltre il dialogo che la macchina instaura con il bambino rende lo studio molto più divertente.
In realtà, My Talking Computer non è un vero e proprio computer, poiché non può essere programmato dall'utente, ma applica la tecnologia dei microcomputer ad una particolare area didattica. Tuttavia, essendo notevolmente sofisticato, può essere un valido supporto per altre applicazioni dello stesso genere.

MY TALKING COMPUTER

DIMENSIONI

mm. 250 x 230 x 70

SOFTWARE

Il software è fornito su ROM, interne al computer. Altri programmi sono disponibili su moduli ROM aggiuntivi, da inserire lateralmente all'apparecchio

PRO

Il My Talking Computer è estremamente facile da usare. La semplicità di caricamento e di avvio dei programmi, assieme alle capacità vocali dell'apparecchio ne permettono l'impiego anche a bambini che non sanno ancora leggere

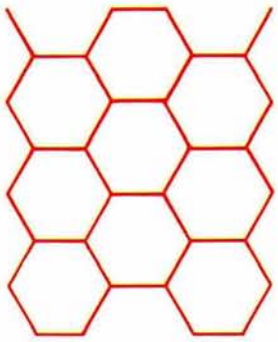
CONTRO

La pronuncia di alcune parole sintetizzate risulta leggermente indistinta. L'apparecchio non è programmabile nel vero senso della parola, quindi non si può definire un vero e proprio computer

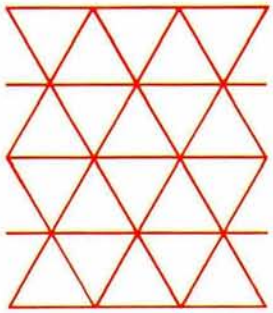
Traguardo finale

In quest'ultima parte dedicata al LOGO vediamo come creare figure "tassellate"

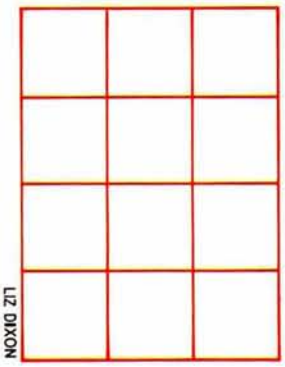
Tre esempi di tassellature



Moduli a esagono



Moduli a triangolo equilatero



LIZ DIXON

Moduli a quadrato

Solo alcuni poligoni regolari permettono di creare delle "tassellature" (vedere figure a margine): i triangoli equilateri, i quadrati e gli esagoni. Nessun altro poligono si adatta infatti a questa tecnica. Consideriamo, ad esempio, il pentagono. Ciascun angolo interno di un pentagono regolare misura 108°. Affiancando tre di questi angoli si ottiene un angolo di 324°: resta quindi uno spazio non coperto di 360°. Aggiungendo un quarto angolo otteniamo un angolo totale di 432°, con ovvia sovrapposizione di due pentagoni. Al contrario le tassellature dei quadrati, dei triangoli equilateri e degli esagoni combaciano perfettamente perché gli angoli di questi poligoni (rispettivamente 90°, 60° e 120°) sono tutti sottomultipli di 360°.

È abbastanza semplice sviluppare procedure LOGO per disegnare le tassellature. Probabilmente il sistema migliore è quello di scrivere una procedura trasparente (vedere a pag. 1044) che traccia la figura, o motivo iniziale, partendo dal centro per poi tornarvi. Questa procedura viene poi chiamata da una "superprocedura", che sposta ripetutamente la tartaruga al centro della successiva figura da tracciare. Le seguenti procedure tracciano una tassellatura composta da quadrati. Le tassellature dei triangoli equilateri e degli esagoni seguono lo stesso procedimento.

```
TO TASSELLA
   PENUP SETXY (—100)(90) PENDOWN
   REPEAT 5 [LINEAVERT SETXY (—100)(YCOR — 40)]
END

TO LINEAVERT
   REPEAT 5 [QUADRATO 40 SETX XCOR + 40]
END

TO QUADRATO :L
   PENUP LEFT 45
   FORWARD :L * (SQRT 2) / 2
   RIGHT 135 PENDOWN
   FORWARD :L RIGHT 90 FORWARD :L RIGHT 90
   FORWARD :L RIGHT 90 FORWARD :L RIGHT 90
   PENUP LEFT 135
   BACK :L * (SQRT 2) / 2 RIGHT 45
END
```

I motivi che compongono le tassellature possono essere più complessi di semplici figure geometriche e possono essere costruiti fondendo tra loro figure diverse. Le tassellature basate su poligoni regolari offrono però più possibilità; infatti la maggior parte delle creazioni di M.C. Escher, il padre di questa tecnica, sono variazioni di tassellature regolari.

Per realizzare configurazioni più complesse basta modificare la procedura QUADRATO, sostituendo i comandi FORWARD che tracciano i lati del quadrato con apposite procedure. L'unica regola da rispettare è che ad ogni modifica del lato superiore del quadrato deve corrispondere un'identica modifica della base: lo stesso principio vale per gli altri due lati del quadrato. In questo modo è possibile ottenere motivi che combaciano perfettamente.

Per disegnare il motivo base si devono sostituire i comandi che tracciano i lati del quadrato con procedure che tracciano nuove linee. La tassellatura riprodotta in basso viene realizzata chiamando nuovamente TASSELLA, dopo aver definito queste procedure:

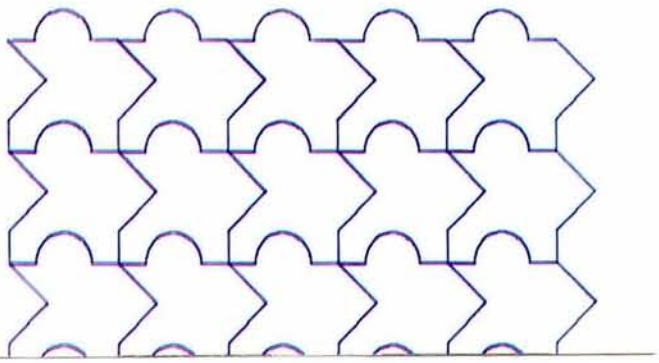
```
TO QUADRATO :L
   PENUP LEFT 45
   FORWARD :L * (SQRT 2) / 2
   RIGHT 135 PENDOWN
   LATOA :L RIGHT 90 LATOB :L RIGHT 90
   LATOC :L RIGHT 90 LATOD :L RIGHT 90
   PENUP LEFT 135
   BACK :L * (SQRT 2) / 2
   RIGHT 45
END

TO LATOA :L
   FORWARD :L / 4 LEFT 90
   REPEAT 19 [FORWARD 2 * 3.1416 * :S / 144 RIGHT 10]
   LEFT 100 FORWARD :L / 4
END

TO LATOB :L
   LEFT 45 FORWARD :L / 2
   RIGHT 90 FORWARD :L / 2
   LEFT 45 FORWARD :L — :L * (SQRT 2) / 2
END

TO LATOC :L
   FORWARD :L / 4 RIGHT 90
   REPEAT 19 [FORWARD 2 * 3.1416 * :L / 144 LEFT 10]
   RIGHT 100 FORWARD :L / 4
END

TO LATOD :L
   FORWARD :L — :L * (SQRT 2) / 2 RIGHT 45
   FORWARD :L / 2 LEFT 90
   FORWARD :L / 2 RIGHT 45
END
```



Naturalmente la tecnica adottata da Escher era più complessa, in quanto egli modificava anche il motivo base via via che la tassellatura cresceva. Una procedura LOGO di questo tipo sarebbe molto interessante...

La tecnica del "mosaico"

Nel loro volume di successo *Turtle Geometry*, Harold Abelson e Andrea di Sessa si sono avvicinati al tema delle tassellature in modo originale. Hanno pensato di tracciare una figura all'interno di un quadrato (ad esempio un motivo di due linee su una mattonella), poi hanno combinato quattro di questi quadrati ottenendone uno di dimensioni maggiori. Quindi hanno ripetuto la procedura col nuovo quadrato e così via.
Esistono varie tecniche per creare "mosaici" di questo genere. In effetti, poiché ciascuno dei quattro quadrati può essere orientato nelle quattro direzioni, sono possibili 256 diverse combinazioni (4 x 4 x 4 x 4). La configurazione ottenuta dipende dal motivo base della mattonella e dal metodo impiegato per realizzare il mosaico. La procedura MBASE, traccia il motivo rappresentato sulla destra.

```
TO MBASE :L
   PENDOWN LEFT 45
   FORWARD :L * (SQRT 2) / 2
   BACK :L * (SQRT 2) / 2
   LEFT 90 FORWARD :L * (SQRT 2) / 2 PENUP
   BACK :L * (SQRT 2) PENDOWN
   FORWARD :L * (SQRT 2) / 4 LEFT 45
   FORWARD :L / 2 LEFT 45
   FORWARD :L * (SQRT 2) / 4 PENUP
   BACK :L * (SQRT 2) / 2 LEFT 135
END
```

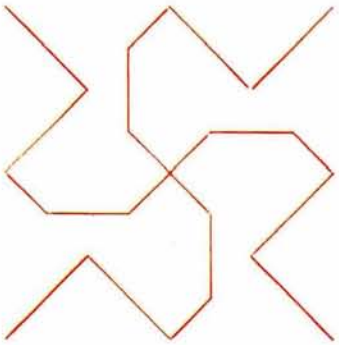
La seguente procedura MOTIVO1 traccia il motivo del mosaico rappresentato più sotto. Per eseguirla occorre digitare DRAW PENUP MOTIVO1 [MBASE] 100.



IAN McKINNELL sul MACINTOSH

Figure in trasformazione
I motivi di una tassellatura combaciano perfettamente tra loro e ricoprono per intero il piano. L'opera "Metamorfosi II" di M.C. Escher, che ha ispirato questa immagine creata da un computer, dimostra come sia possibile modificare progressivamente un semplice motivo base, come un esagono regolare, per ottenere tassellature più complesse, in questo caso l'articolata immagine di una lucertola. La tecnica di Escher evidenzia la graduale trasformazione da semplice a complesso via via che la tassellatura si sviluppa

```
TO MOTIVO1 :PROC :L
   QUADR 0 :PROC :L
   QUADR 0 :PROC :L
   QUADR 0 :PROC :L
   QUADR 0 :PROC :L
END
```



QUADR traccia uno dei quattro quadrati che compongono il quadrato più grande. La procedura richiede tre parametri in ingresso: :A indica l'orientazione del quadrato, :PROC è il nome della procedura che traccia il motivo e :L è la lunghezza del lato. Considerando le limitate dimensioni dello schermo, non è possibile continuare ad espandere le misure del quadrato, quindi la procedura assume come parametro le dimensioni del più grande quadrato visualizzabile e su questa base calcola le dimensioni dei quadrati più piccoli.

```
TO QUADR :A :PROC :L
   FORWARD :L / 4 RIGHT 90
   FORWARD :L / 4 RIGHT 90 -* :A
   RUN SENTENCE :PROC :L / 2 LEFT 90 -* :A
   BACK :L / 4 LEFT 90
   BACK :L / 4 RIGHT 90
END
```

Le seguenti procedure tracciano altri possibili motivi:

```
TO MOTIVO2 :PROC :L
   QUADR 0 :PROC :L
   QUADR 1 :PROC :L
   QUADR 2 :PROC :L
   QUADR 3 :PROC :L
END

TO MOTIVO3 :PROC :L
   QUADR 0 :PROC :L
   QUADR 2 :PROC :L
   QUADR 3 :PROC :L
   QUADR 1 :PROC :L
END

TO MOTIVO4 :PROC :L
   QUADR 3 :PROC :L
   QUADR 2 :PROC :L
   QUADR 1 :PROC :L
   QUADR 0 :PROC :L
END

TO MOTIVO5 :PROC :L
   QUADR 2 :PROC :L
   QUADR 1 :PROC :L
   QUADR 0 :PROC :L
   QUADR 3 :PROC :L
END
```

MBASE, ossia la procedura che traccia il motivo base, richiede un solo parametro. Nella linea di comando MOTIVO1 [MBASE] 100, la parte MOTIVO1 [MBASE] si può considerare un comando che richiede solo la presenza di un numero per tracciare un motivo. Quindi possiamo usare [MOTIVO1 [MBASE]] come dato in ingresso per un'altra procedura MOTIVO:

```
MOTIVO2 [MOTIVO1 [MBASE]] 100
```

Questa può essere ancora sviluppata:

```
MOTIVO3 [MOTIVO2 [MOTIVO1 [MBASE]]] 100
```

Otteniamo così un mosaico a tre livelli con 256 x 256 x 256 combinazioni possibili e nulla vieta di introdurre un numero di livelli ancora maggiore!



Logo & Logo

In tutte le versioni LCSI, usare SETPOS, seguito da una lista, al posto di SETXY. Nel LOGO Atari, usare l'abbreviazione SE al posto di SENTENCE

IDEE... IN LIBERTÀ!

Questi sono altri esempi di possibili mosaici:



MOTIVO1 [MOTIVO4 [MOTIVO5 [MBASE]]] 100



MOTIVO4 [MOTIVO3 [MOTIVO3 [MBASE]]] 100

Missione di ricognizione

Continuano gli esperimenti con il veicolo robot realizzato in "Laboratorio"

Affinché un robot possa esplorare una data area, è necessario ideare un sistema di ricognizione che copra con precisione l'intera superficie. Un buon modello di ricognizione potrebbe essere quello di un robot che si muove metodicamente avanti e indietro alla ricerca di un oggetto: localizzatone uno, ne esplora tutti i lati prima di riprendere la ricognizione dell'area.
Purtroppo un simile algoritmo presenta notevoli difficoltà di programmazione e possono sorgere dei problemi quando l'area contiene più oggetti. In alternativa, dopo avere incontrato l'oggetto, il robot può invertire la propria direzione e riprendere la ricognizione senza esplorarlo. Le fasi di questo secondo algoritmo si possono descrivere nel modo seguente:

```
RIPETI
    RIPETI
    Avanza alla ricerca di un oggetto
    FINCHÉ incontra un margine dell'area da esplorare o incontra un oggetto
Si sposta sulla successiva "striscia" di esplorazione
Esegue una rotazione e si muove
FINCHÉ non raggiunge il margine opposto dell'area
```

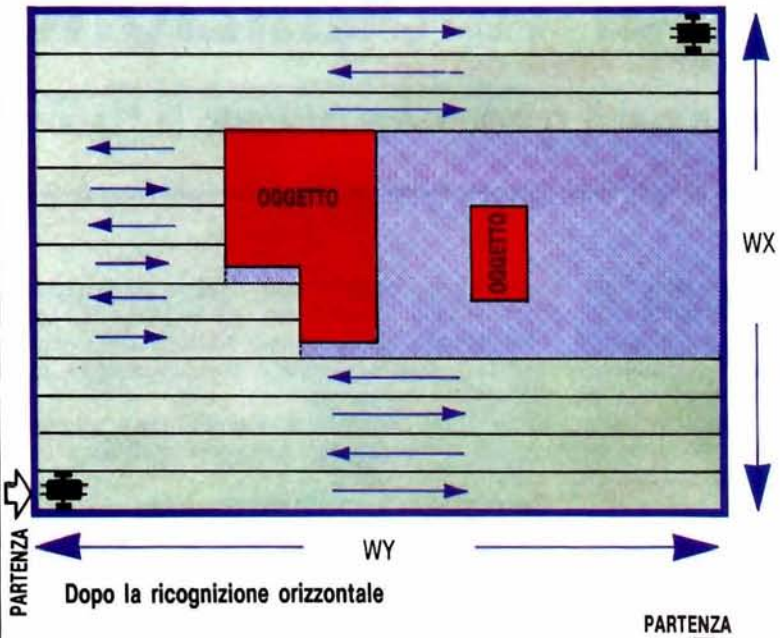
Modifica dei sensori

Il robot può effettuare una ricognizione precisa solo dopo una leggera modifica ai due sensori di contatto frontali. Infatti, dato che le sue ruote sporgono su entrambi i lati, può accadere che urtino un oggetto quando il robot passa nelle sue vicinanze. Per eliminare questo inconveniente è necessario fissare su entrambi i sensori frontali una piastrina a protezione delle ruote. La piastrina deve essere rettangolare (circa 95 x 25 mm.) e di materiale rigido, come metallo o plastica. Il materiale usato non deve tuttavia essere troppo pesante, altrimenti potrebbe far scattare i microinterruttori dei sensori. Le piastrine vengono incollate o fissate con nastro adesivo e vanno posizionate in modo da coprire esattamente la sezione delle ruote. A lavoro ultimato, verificare il corretto funzionamento dei sensori.

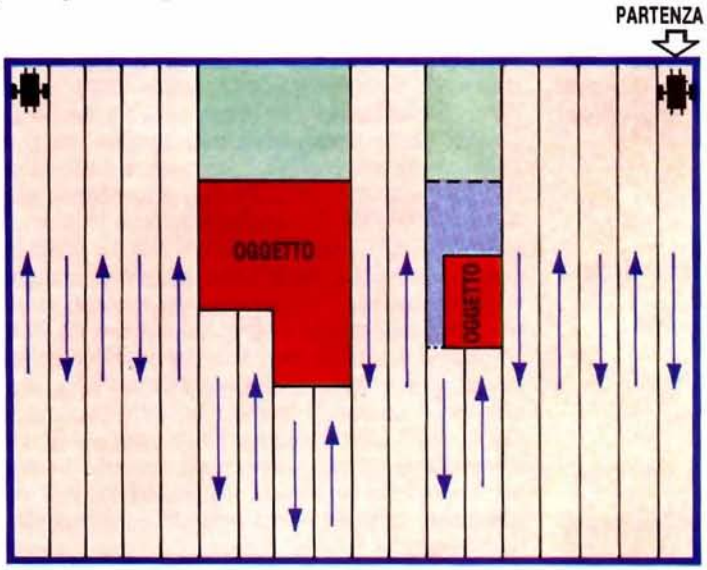
Programma di ricognizione dell'oggetto

Il programma che consente al robot di esplorare l'area esegue una doppia ricognizione orizzontale e verticale. Al termine di ogni "traversata", il computer traccia sullo schermo una fascia colorata corrispondente alla superficie esplorata.
Alla fine dell'intera ricognizione avremo quindi un'immagine completa dell'area esaminata dal robot.
Entrambe le versioni del programma chiedono all'utente di inserire le misure dell'area in esame. Usando lo schermo ad alta risoluzione del Commodore 64 o il MODE 4 del Micro BBC, ad ogni pixel acceso sullo schermo corrisponde una superficie di 4 mm², quindi le misure massime della base e dell'altezza dell'area sono rispettivamente 1279 e 1023 mm.. Ricordiamo che entrambi i programmi usano fasce d'esplorazione larghe 40 mm..
Dopo ogni traversata, l'immagine sullo schermo viene aggiornata da due subroutine chiamate XPLOT e YPLOT. Per visualizzare le fasce, la versione BBC usa i comandi MOVE e DRAW che tracciano una serie di linee parallele adiacenti. La versione Commodore 64 ricorre invece alle due routine in codice macchina PLOTSUB e LINESUB, progettate in precedenza (vedere alle pagg. 818 e 899), che suppliscono alla mancanza di comandi grafici in alta risoluzione del BASIC Commodore. I file oggetto di queste routine vengono caricati usando le linee 30 e 40 della versione Commodore del programma, oppure con i relativi programmi di caricamento BASIC (in questo caso possono essere omesse le linee 30 e 40). Entrambe le routine devono essere caricate ed eseguite separatamente, prima di caricare e lanciare il programma principale.
Al termine delle due ricognizioni può accadere che il programma lasci inesplorata una parte consistente dell'area. Inoltre possono verificarsi altre difficoltà, poiché il successo della ricognizione dipende sia dalla forma degli oggetti sia dalla loro disposizione. I migliori risultati si ottengono con figure rettangolari orientate secondo gli assi orizzontale e verticale. Le figure triangolari e circolari costringono il robot a lasciare ampie zone inesplorate, perché il programma tende a "squadrare" le irregolarità degli oggetti incontrati. Questo inconveniente può essere in parte eliminato introducendo una seconda ricognizione verticale. Infine il robot può trovarsi in difficoltà se incontra un oggetto passando da una fascia d'esplorazione alla successiva. Infatti, durante tale operazione, non viene eseguita alcuna verifica sullo stato dei sensori.

AREE "CIECHE"



Dopo la ricognizione orizzontale



Dopo la ricognizione verticale

Chiavi:

Fascia di esplorazione orizzontale

Fascia di esplorazione verticale

Area "cieca"

Il sistema più semplice per esplorare un'area rettangolare è quello di eseguire due ricognizioni tra loro perpendicolari. Quando il robot incontra un oggetto o raggiunge il margine dell'area, passa alla fascia successiva e prosegue l'esplorazione nella direzione opposta.
Purtroppo con questa tecnica alcune aree situate dietro agli oggetti risultano "cieche"; tuttavia con la seconda ricognizione è possibile rivelare molti dei particolari sfuggiti alla prima analisi. La relativa imprecisione del processo dipende dalla larghezza della fascia di esplorazione usata e dalla reciproca orientazione degli oggetti.

KEVIN JONES

LISTATO MICRO BBC

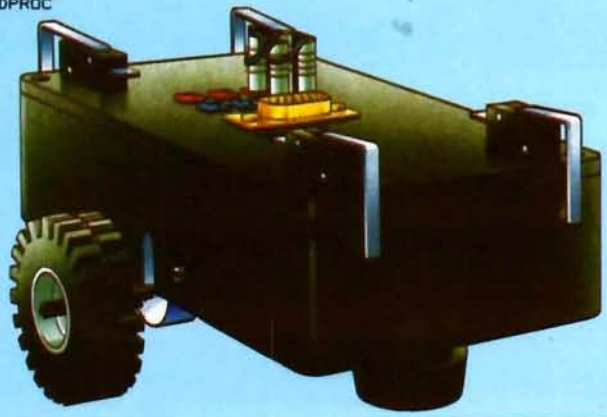
```
 10 REM **** RICERCA OGGETTO  BBC ****
 20 MODE 4
 30 PROCinizializza
 40 PROCdim_area
 50 PROCesploraz_orizz
 60 PROCesploraz_vert
 70 ENDPROC
 80 DEF PROCesploraz_orizz
 90 bersaglio=wx: senso=sin: xiniz=0
100 REPEAT
110 REPEAT
120 PROCmuove(avanti,dx): x=x+dx
130 UNTIL (?REGDAT AND 192)<>nessun_sensore OR x=bersaglio
140 PROCxplot
150 dx=-dx: y=y+dy: xiniz=x
160 IF bersaglio=0 THEN bersaglio=wx ELSE bersaglio=0
170 IF y<wy THEN PROCfascia_succ(senso)
180 IF senso=des THEN senso=sin ELSE senso=des
190 UNTIL y>=wy
200 ENDPROC
210 :
220 PROCesploraz_vert
```

LISTATO COMMODORE 64

```
 10 REM **** RICERCA OGGETTO C64 ****
 20 DN=8: REM PER CASSETTA DN=1
 30 IF A=0 THEN A=1: LOAD "PLOTSUB.HEX",DN,1
 40 IF A=1 THEN A=2: LOAD "LINESUB.HEX",DN,1
 50 GOSUB 1000: REM INIZIALIZZA
 60 GOSUB 2000: REM DIMENSIONI DELL'AREA
 70 GOSUB 5000: REM ABILITA IL MODO DI ALTA RISOLUZIONE
 80 GOSUB 3000: REM ESPLORAZ. ORIZZONTALE
 90 GOSUB 4000: REM ESPLORAZ. VERTICALE
100 GOSUB 5100: REM DISABILITA L'ALTA RISOLUZIONE
110 END
120 :
1000 REM **** INIZIALIZZA ****
1010 DDR=56579:REGDAT=56577
1020 POKE DDR,15:POKE REGDAT,1
1030 AV=4:IN=2:SI=6:DE=0
1040 PD=3.14446:PA=375/90
1050 RB=128:LB=64:BB=0:NB=192
1060 WD=40:DW=60/10:X=0:Y=0:DX=DW:DY=WD
1070 REM ** INDIRIZZI INIZIALI M/C **
1080 HIRES=49422:LINESUB=49934
1090 RETURN
```

segue listato Micro BBC

```
230 IF x=0 THEN senso=sin: dx=ampiezza ELSE senso=des:
    dx=-ampiezza
240 dy=-dw: bersaglio=0: yiniz=y
250 PROCruota(senso,90)
260 REPEAT
270 REPEAT
280 PROCmuove(avanti,dy): y=y+dy
290 UNTIL (?REGDAT AND 192)<>nessun_sensore OR y=bersaglio
300 PROCyplot
310 dy=-dy: x=x+dx: yiniz=y
320 IF bersaglio=0 THEN bersaglio=wy ELSE bersaglio=0
330 PROCfascia_succ (senso)
340 IF senso=des THEN senso=sin ELSE senso=des
350 UNTIL x>=wx OR x<0
360 ENDPROC
370 :
380 DEF PROCfascia_succ(movim)
390 PROCmuove(indietro,30)
400 PROCruota(movim,90)
410 PROCmuove(avanti,ampiezza)
420 PROCruota(movim,90)
430 ENDPROC
440 :
450 DEF PROCdim_area
460 INPUT "BASE X IN mm.";wx
470 IF wx>1279 THEN 460
480 INPUT "ALTEZZA Y IN mm.";wy
490 IF wy>1023 THEN 480
500 wx=ampiezza*(wx DIV ampiezza)
510 wy=ampiezza*(wy DIV ampiezza)
520 CLS
530 ENDPROC
540 :
550 DEF PROCinizializza
560 DDR=&FE62: REGDAT=&FE60
570 ?DDR=15: REM LINEE 0-3 IN USCITA
580 ?REGDAT=1: REM ACCENDE IL BIT DI AZZERAMENTO
590 avanti=4: indietro=2: sin=6: des=0
600 rapp_id=3.34446: rapp_ia=375/90
610 sensore_des=128: sensore_sin=64
620 entrambi_sensori=0: nessun_sensore=192
630 ampiezza=40: dw=ampiezza/10
640 x=0: y=0: dx=dw: dy=ampiezza
650 MOVE 0,0
660 ENDPROC
670 :
680 DEF PROC(senso,distanza)
690 ?REGDAT=(?REGDAT AND 1) OR senso
700 impulsi=rapp_id*ABS(distanza)
710 FOR I=1 TO impulsi: PROCimpulso: NEXT I
720 ENDPROC
730 :
740 DEF PROCruota(senso,angolo)
750 ?REGDAT=(?REGDAT AND 1) OR senso
760 impulsi=rapp_ia*angolo
770 FOR I=1 TO impulsi: PROCimpulso: NEXT I
780 ENDPROC
790 :
800 DEF PROCimpulso
810 ?REGDAT=(?REGDAT OR 8)
820 ?REGDAT=(?REGDAT AND 247)
830 ENDPROC
840 DEF PROCxplot
850 FOR I=0 TO ampiezza STEP 4
860 MOVE xiniz,y+I
870 DRAW x,y+I
880 NEXT I
890 ENDPROC
900 :
910 DEF PROCyplot
920 FOR I=0 TO ampiezza STEP 4
930 MOVE x+I,yiniz
940 DRAW x+I,y
950 NEXT I
960 ENDPROC
```



segue listato Commodore 64

```
1100 :
2000 REM **** DIMENSIONI DELL'AREA ****
2010 INPUT "BASE X IN mm.";WX
2020 IF WX/4>319 THEN 2010
2030 INPUT "ALTEZZA Y IN mm.";WY
2040 IF WY/4>199 THEN 2030
2050 WX=WD*INT(WX/WD):WY=WD*INT(WY/WD)
2060 RETURN
2070 :
3000 REM **** ESPLORAZ. ORIZZONTALE ****
3010 T=WX:SE=SI:XS=0
3020 DR=AV: DS=DX: GOSUB 7000: REM MUOVE
3030 X=X+DX
3040 IF(PEEK(REGDAT)AND 192)=NB AND X<>TG THEN 3020
3050 GOSUB 9000: REM S/R XPLOT
3060 DX=-DX:Y=Y+DY:XS=X
3070 IF TG=0 THEN TG=WX:GOTO 3090
3080 TG=0
3090 IF Y<WY THEN WA=SE: GOSUB 8000: REM BANDA SUCC.
3100 IF SE=DE THEN SE=SI:GOTO3120
3110 SE=DE
3120 IF Y<WY THEN 3020
3130 RETURN
3140 :
4000 REM **** ESPLORAZ. VERTICALE ****
4010 IF X=0 THEN SE=SI:DX=WD:GOTO 4030
4020 SE=DE:DX=-WD
4030 DY=-DY:TG=0:YS=Y
4040 DR=SE: AG=90: GOSUB 7100: REM RUOTA
4050 DR=AV: DE=DY: GOSUB 7000: REM MUOVE
4060 Y=Y+DY
4070 IF (PEEK(REGDAT)AND 192)=NB AND Y<>TG THEN 4050
4080 GOSUB 9100: REM S/R YPLOT
4090 DY=-DY:X=X+DX:YS=Y
4100 IF TG=0 THEN TG=WY:GOTO 4120
4110 TG=0
4120 WA=SE: GOSUB 8000: REM BANDA SUCC.
4130 IF SE=DE THEN SE=SI:GOTO 4150
4140 SE=DE
4150 IF X<WX AND X>0 THEN 4050: REM RIPETE
4160 RETURN
4170 :
5000 REM **** ABILITA L'ALTA RISOLUZIONE ****
5010 POKE 49408,1:POKE 49409,1
5020 POKE 49410,1:SYS HIRES:RETURN
5030 :
5100 REM **** DISABILITA L'ALTA RISOLUZIONE ****
5110 POKE 49408,0:POKE 49409,0
5120 POKE 49410,1:SY S HIRES:RETURN
5130 :
6000 REM **** LANCIA LA ROUTINE LINESUB ****
6010 MHI=INT(X1/256):MLO=X1-256*MHI
6020 NHI=INT(X2/256):NLO=X2-256*NHI
6030 POKE 49920,MLO:POKE 49921,MHI
6040 POKE 49922,NLO:POKE 49923,NHI
6050 POKE 49924,Y1:POKE 49925,Y2
6060 SYS LINESUB:RETURN
6070 :
7000 REM **** MUOVE (DR,DS) ****
7010 POKE REGDAT,(PEEK(REGDAT) AND 1)OR DR
7020 PL=PD*DS
7030 FOR I=1 TO PL:GOSUB7200:NEXT I
7040 RETURN
7050 :
7100 REM **** RUOTA (DR,AG) ****
7110 POKE REGDAT,(PEEK(REGDAT) AND 1)OR DR
7120 PL=PA*DS
7130 FOR I=1 TO PL:GOSUB7200:NEXT I
7140 RETURN
7150 :
7200 REM **** IMPULSO ****
7210 POKE REGDAT,PEEK(REGDAT) OR 8
7220 POKE REGDAT,PEEK(REGDAT) AND 247
7230 RETURN
7240 :
8000 REM **** BANDA SUCCESSIVA ****
8010 DR=IN: DS=30: GOSUB 7000: REM MUOVE
8020 DR=WA: AG=90: GOSUB 7100: REM RUOTA
8030 DR=AV: DS=WD: GOSUB 7000: REM MUOVE
8040 DR=WA: AG=90: GOSUB 7100: REM RUOTA
8050 RETURN
9000 REM **** S/R XPLOT ****
9010 FOR I=0 TO WD
9020 X1=XS/4:Y1=(Y+1)/4:X2=X/4:Y2=Y1
9030 GOSUB 6000: REM LANCIA LA ROUTINE LINESUB
9035 NEXT I
9040 RETURN
9100 REM **** S/R YPLOT ****
9110 FOR I=0 TO WD
9120 X1=(X+1)/4:Y1=YS/4:X2=X1:Y2=Y/4
9130 GOSUB 6000: REM LANCIA LA ROUTINE LINESUB
9135 NEXT I
9140 RETURN
```

Le chiamate OSWORD del BBC

Prosegue l'esame delle chiamate al sistema operativo del BBC

Le chiamate OSBYTE, descritte in precedenza, hanno un grosso limite: non possono accettare più di due parametri (uno nel registro X e uno nel registro Y). Spesso però due soli parametri non sono sufficienti per controllare le routine del sistema operativo. Le chiamate OSWORD, che prevedono il passaggio di un numero superiore di parametri, si dimostrano un'utile alternativa alle OSBYTE.
OSWORD utilizza un'area di memoria, chiamata "blocco dei parametri", per scambiare i parametri con il sistema operativo. Quest'area di memoria, le cui dimensioni variano a seconda della chiamata OSWORD selezionata, può trovarsi in qualsiasi settore della RAM per l'utente del computer. Quando viene chiamata una routine di OSWORD, il valore contenuto nel registro A indica quale routine è richiesta, mentre i registri X e Y forniscono l'indirizzo del primo byte del blocco di parametri: X contiene il byte basso dell'indirizzo, Y quello alto. Quindi, se il blocco di parametro è situato all'indirizzo &0A00, i registri X ed Y contengono rispettivamente i valori 00 e &0A:

Codice Macchina	BASIC
LDX #0 LDY #&0A	X%=0 Y%=&0A

L'indirizzo di chiamata per le routine OSWORD è &FFF1. Come le chiamate OSBYTE, anche le OSWORD usano un vettore, il cui indirizzo è &20C. Così, per chiamare una routine OSWORD quando il registro A contiene il valore 1, si devono inserire i parametri nel blocco di parametri e chiamare la routine desiderata con il programma (blocco_parametri rappresenta l'indirizzo del primo byte del blocco):

```
LDX   #blocco_parametri MOD 256
LDY   #blocco_parametri DIV 256
LDA   #1
JSR   &FFF1
```

Naturalmente i valori inseriti nel blocco di parametri variano a seconda della chiamata OSWORD fatta.

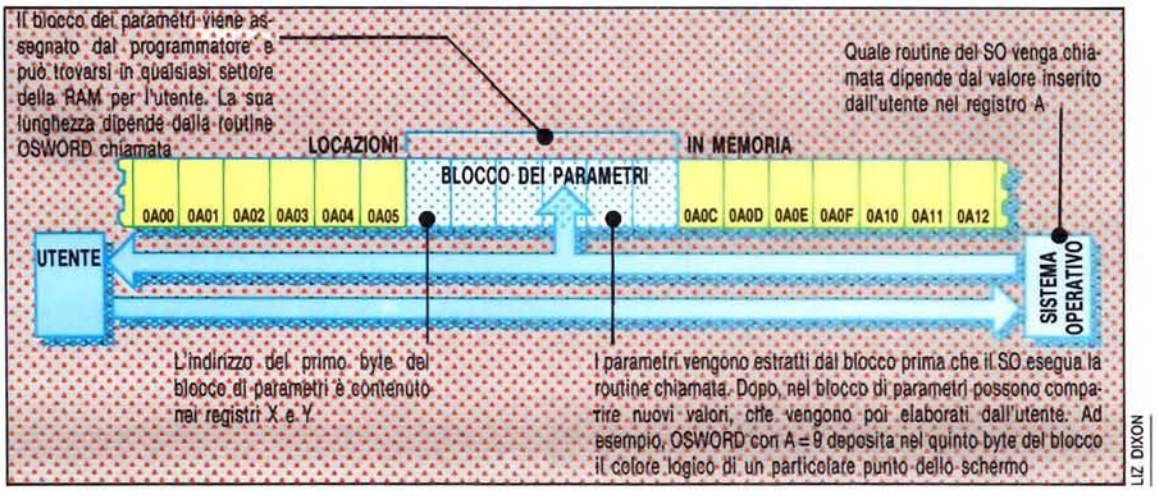
Funzioni delle chiamate OSWORD

Le chiamate OSWORD permettono di leggere informazioni dal flusso di dati in ingresso e depositarli in memoria, di eseguire comandi SOUND e ENVELOPE, di accedere al sistema di archiviazione su disco e di compiere molte altre operazioni. Le diverse routine OSWORD vengono indicate con nomi che si riferiscono al valore contenuto nel registro A quando viene chiamato l'indirizzo &FFF1. Ad esempio, OSWORD con A=0.

• OSWORD *con A=0:* questa chiamata permette di leggere una linea di informazioni dal flusso dei dati in ingresso selezionato. Inoltre consente di definire il numero massimo di caratteri in ingresso ed i valori minimo e massimo consentiti per i codici ASCII dei caratteri che vengono introdotti. I dati letti vengono quindi depositati in memoria a partire dall'indirizzo indicato dai primi due byte

Maggiori opportunità

Le routine OSWORD si dimostrano molto vantaggiose nel controllo delle routine del SO, per le quali spesso non sono sufficienti i due parametri offerti dalle routine OSBYTE



del blocco di parametri. Il blocco di parametri viene strutturato nel seguente modo:

Blocco dei parametri di OSWORD con A=0	
Byte	**Funzione**
0	Byte basso dell'indirizzo del buffer
1	Byte alto dell'indirizzo del buffer
2	Numero massimo dei caratteri in ingresso
3	Minimo codice ASCII consentito
4	Massimo codice ASCII consentito

I registri X e Y puntano al byte 0 del blocco. Il seguente programma chiarisce il funzionamento della chiamata. Il blocco di parametri viene predisposto per accettare fino a 7 caratteri, con codici ASCII compresi tra 32 e 90.

```
 10 DIM C 20
 20 A%=0
 30 X%=C MOD 256
 40 Y%=C DIV 256
 50 C?0=&00
 60 C?1=&0A: REM buffer all'indirizzo &0A00
 70 C?2=7
 80 C?3=32
 90 C?4=90
100 CALL &FFF1
110 PRINT $(&0A00)
```

Si lanci il programma e si digitino alcuni caratteri. Il tasto Delete conserva la sua normale funzione, mentre sia Escape interrompono la routine OSWORD. Riguardo a questa chiamata è importante sottolineare che i caratteri digitati, i cui codici ASCII non sono compresi nella gamma consentita, sono visualizzati sullo schermo ma non aggiunti ai caratteri immagazzinati nel buffer in memoria. Se viene digitato un numero di caratteri troppo alto, il computer emette un segnale acustico d'avvertimento ("bip"). In codice macchina, se si preme il tasto Escape per interrompere una chiamata OSWORD, il flag di carry viene posto ad uno, altrimenti esso rimane a 0. Il registro Y conteggia i caratteri inseriti, compreso il "ritorno-carrello".

• OSWORD *con A=1 e A=2:* con queste chiamate il programmatore in codice macchina può accedere alla variabile TIME del BASIC. OSWORD con A=1 consente di leggere il valore corrente di TIME e di copiare i 5 byte che contengono tale valore nel blocco di parametri abbinato alla chiamata, che deve quindi essere lungo almeno 5 byte. Il byte meno significativo di TIME viene registrato nel byte 0 del blocco, quello più significativo nel quinto byte del blocco.

OSWORD con A=2 permette invece di assegnare a TIME un dato valore. Il byte meno significativo di questo valore viene copiato nel primo byte del blocco di parametri, quello più significativo nel quinto byte. Infine viene eseguita la chiamata a OSWORD. Il seguente programma assegna a TIME il valore 100:

```
 10 DIM C 20
 40 A%=2
 50 C?0=100
 60 C?1=0
 70 C?2=0
 80 C?3=0
 90 C?4=0
100 CALL &FFF1
```

• OSWORD *con A=3 ed A=4:* il Micro BBC possiede un secondo orologio (clock) interno, chiamato temporizzatore di intervalli (o di eventi), che non viene controllato dalla variabile TIME bensì dal sistema di eventi del computer (questo sistema viene esaminato in seguito). Questo temporizzatore scandisce il tempo in centesimi di secondo a partire da un valore dato e genera un evento quando raggiunge lo zero. L'evento generato è facilmente controllabile e può essere usato per far eseguire al computer una serie di operazioni dopo un dato intervallo di tempo.

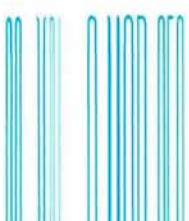
OSWORD con A=3 funziona in modo simile ad OSWORD con A=1 e consente di copiare il valore corrente del temporizzatore di intervalli nel blocco di parametri. OSWORD con A=4 permette invece di assegnare un dato valore al temporizzatore di intervalli (ossia una procedura simile a quella seguita in OSWORD con A=2 per definire la variabile TIME). Quindi, copiando il valore &FFFFFFFC dal temporizzatore nel blocco di parametri, otterremo la comparsa dell'evento ritardato di 0,004 secondi.

• OSWORD *con A=5 ed A=6:* queste sono chiamate più specializzate e si dimostrano molto utili per i programmatori che adoperano Micro BBC dotati di un secondo processore. Poiché solo uno dei due processori si trova all'interno del computer, questo può comunicare col secondo processore attraverso l'interfaccia Tube e usare il computer come un processore di ingresso/uscita (I/O). In un sistema di questo tipo il Micro BBC si comporta come un apparecchio "asservito" ed esegue tutte le operazioni ausiliarie necessarie al secondo processore, come la gestione degli input provenienti dalla tastiera, dello schermo e delle principali funzioni di I/O.

Blocchi dei parametri di OSWORD con A=5 e A=6		
Byte	**OSWORD con A=5**	**OSWORD con A=6**
0	Byte basso dell'indirizzo del proc. di I/O	Byte basso dell'indirizzo del proc. di I/O
1	Indirizzo del byte da copiare (origine)	Indirizzo del byte da scrivere (destinazione)
2	Indirizzo del byte da copiare (origine)	Indirizzo del byte da scrivere (destinazione)
3	Byte alto dell'indirizzo del proc. di I/O	Byte alto dell'indirizzo del proc. di I/O
4	Dopo la chiamata, il byte all'indirizzo specificato	Il byte da scrivere all'indirizzo specificato

La tabella mostra la struttura dei blocchi dei parametri per A=5 (che legge un byte della memoria del processore di I/O) e A=6 (che scrive nella memoria del processore di I/O). Se l'indirizzo da copiare è a 16 bit, come nel caso del processore di I/O di un normale Micro BBC, il suo byte basso viene memorizzato in (XY+0) e quello alto in (XY+1).

• OSWORD *con A=7 e A=8:* con queste chiamate il programmatore può manipolare i suoni





con A=8 simulano i rispettivi comandi SOUND e ENVELOPE del BASIC. Esaminiamo le due chiamate separatamente:

• OSWORD *con A=7:* usa un blocco di parametri di 8 byte (vedere la tabella seguente). Inoltre deve ricevere indicazioni sulla corrispondenza di ciascun elemento del blocco dei parametri con ciascun parametro del comando SOUND.

Blocco dei parametri di OSWORD con A=7	
Byte	**Funzione**
0	Byte basso del NUMERO DI CANALE
1	Byte alto del NUMERO DI CANALE
2	Byte basso dell'ESTENSIONE
3	Byte alto dell'ESTENSIONE
4	Byte basso dell'ALTEZZA
5	Byte alto dell'ALTEZZA
6	Byte basso della DURATA
7	Byte alto della DURATA

Il seguente programma, che simula il comando SOUND 1,-15,100,30, chiarisce il funzionamento della chiamata:

```
 10 DIM C 20
 20 X%=C MOD 256
 30 Y%=C DIV 256
 40 A%=7
 50 C?0=1
 60 C?1=0 REM CHANNEL 1
 70 C?2=-15 MOD 256
 80 C?3=-15 DIV 256
 90 C?4=1000
100 C?5=0
110 C?6=30
115 C?7=0
120 CALL &FFF1
130 END
```

Poiché il comando SOUND del BASIC permette di generare suoni con maggior facilità, questa chiamata viene poco usata in BASIC.

• OSWORD *con A=8:* usa un blocco dei parametri composto da 14 byte. Come nelle altre chiamate, i registri X e Y puntano al blocco dei parametri: il primo byte contiene il numero di ENVELOPE, mentre i 13 successivi corrispondono agli altri parametri di inviluppo. Rispetto al corrispondente comando BASIC, questa chiamata consente di controllare con molta più facilità l'inviluppo di un suono.

Chiamate grafiche OSWORD

Byte	Valore
0	2
1	4
2	0
3	0
4	0

• OSWORD *con A=9:* usata all'interno di un programma in codice macchina, produce gli stessi effetti del comando POINT del BASIC. Al ritorno dalla chiamata, il quinto byte del blocco dei parametri contiene il colore logico del punto dello schermo specificato, oppure il valore 255 se il punto specificato non si trova sullo schermo. La struttura del blocco dei parametri per la chiamata OSWORD con A=9 è:

Blocco dei parametri di OSWORD con A=9	
Byte	**Funzione**
0	Byte basso della coordinata X
1	Byte alto della coordinata X
2	Byte basso della coordinata Y
3	Byte alto della coordinata Y
4	Colore logico

• OSWORD *con A=11:* questa chiamata permette di esaminare il colore specificato nella serie di comandi VDU 19 e di individuare il colore "fisico" associato a un dato colore logico: è molto utile quando non si riescono a ricordare i colori precedentemente selezionati per una particolare chiamata VDU 19. Dopo aver eseguito la chiamata, che usa un blocco di parametri di cinque byte, il primo byte del blocco contiene il valore del colore logico ricercato. Per meglio chiarire il suo funzionamento, supponiamo di aver già eseguito il comando VDU 19,2,4,0,0,0. La chiamata OSWORD con A=11 e un valore 2 nel primo byte del blocco dei parametri produce come risultato i valori contenuti nella tabella a margine.

• OSWORD *con A=12:* questa chiamata è solo una versione più veloce del comando VDU 19.

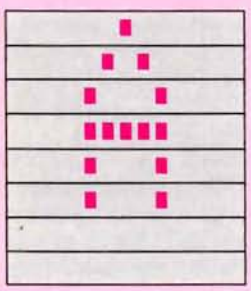
• OSWORD *con A=13:* questa chiamata si dimostra utile solo in casi particolari. Permette infatti di recuperare le coordinate X e Y degli ultimi due punti "visitati" dal cursore grafico.

Questa breve panoramica mette in luce la grande versatilità delle chiamate OSWORD: grazie ad esse l'accesso alle funzioni del Micro BBC risulta molto facilitato.

OSWORD CON A=10

Questa chiamata viene usata spesso per visualizzare caratteri di grosse dimensioni, poiché permette di estrarre la definizione di un qualsiasi carattere nei MODE da 0 a 6. Alla chiamata occorre un blocco di parametri composto da 9 byte: per il suo indirizzamento vengono usati i registri X e Y. Il primo byte del blocco contiene il primo byte del carattere

Blocco dei parametri di OSWORD con A=10	
Byte	**Funzione**
0	Carattere da visualizzare
1	Prima linea di definizione
2	Seconda linea
3	Terza linea
4	Quarta linea
5	Quinta linea
6	Sesta linea
7	Settima linea
8	Ultima linea di definizione



I valori depositati dalla chiamata nel blocco di parametri sono la rappresentazione binaria della definizione del carattere: 1 indica un pixel acceso 0 un pixel spento

I Bulletin Board

Un centro di comunicazione e di scambio aperto a tutti gli appassionati



STEVE CROSS

Il primo bulletin board ha avuto origine in un computer club statunitense, quando i soci decisero che la tradizionale bacheca, ove affiggere messaggi, non era più adeguata alle nuove tecnologie disponibili. Alcuni membri del club pensarono quindi di realizzare un sistema computerizzato, accessibile tramite le normali linee telefoniche.
Attualmente il numero di bulletin board esistenti al mondo, tramite i quali avvengono scambi di messaggi o di software, è incalcolabile. Alcuni sono gestiti da associazioni, altri da semplici hobbisti.
L'accesso alla maggior parte dei "BB" è libero e gratuito (salvo preoccupanti aumenti della bolletta telefonica!): basta comporre il numero di telefono, digitare al computer il proprio nome e fornire dettagli circa il proprio terminale.
In Gran Bretagna, il paese europeo dove i BB si sono diffusi di più, esistono due sistemi principali: The Bread Board System (TBBS), il cui uso e adattabilità sono notevoli e il Central Bulletin Board System (CBBS). Quest'ultimo provvede automaticamente a passare l'elenco degli utenti a tutti i BB collegati, quindi, accedendo anche a uno solo di essi, gli estremi della registrazione (in particolare le specifiche del terminale) vengono memorizzati anche in tutti gli altri BB, semplificandone l'accesso. Per quanto concerne i BB gestiti da hobbisti, il sistema più in voga (sempre in Gran Bretagna), è il TBBS, che è suddiviso in 4 aree principali: messaggi generici, gruppi

Ci sono messaggi per me?

I bulletin board computerizzati offrono ai propri utenti una vasta scelta di servizi, inclusi lo scambio di messaggi generici, informazioni tecniche sui nuovi prodotti, alcuni giochi e un sistema di posta elettronica. Lo scopo principale è quello di offrire a tutti gli utenti un canale di interscambio il più ampio possibile

d'interesse speciale (Special Interest Groups, o SIG), posta elettronica e trasferimento di file.
La sezione messaggi generici è riservata, appunto, allo scambio di messaggi, il cui contenuto non concerne argomenti specifici, ma riguarda in genere richieste tecniche (e, con un po' di fortuna, le relative risposte, depositate da altri utenti), commenti su nuovi apparecchi, notizie dai club, recensioni varie e, spesso, anche barzellette.
La sezione dei SIG, al contrario, è ulteriormente suddivisa in sottosezioni, ciascuna dedicata ad un particolare apparecchio (BBC, CP/M, Tandy, Commodore, Modem, Mercatino dell'usato, ecc.). Se un utente desidera creare un nuovo SIG, basta che depositi una richiesta in tal senso al "sysop", ossia all'operatore di sistema: se il SIG incontra un sufficiente interesse, il sysop ne cura la gestione.
La posta elettronica, spesso chiamata col termine "E-mail", consente lo scambio di corrispondenza di carattere privato tra i vari utenti. I messaggi di questo genere sono accessibili (e pertanto leggibili) soltanto dall'autore, dal destinatario ed eventualmente dall'operatore del sistema, in quanto occorre una speciale "parola chiave", che viene scelta e immessa dall'utente in occasione del primo accesso al sistema e richiesta in seguito ad ogni nuovo accesso.
La sezione dedicata allo scambio di software permette agli utenti sia di depositare sia di prelevare programmi di pubblico dominio (ossia non coperti da copyright).
Non è previsto alcun compenso per i programmi depositati, né alcuna tariffa per quelli prelevati.
Come la sezione per i SIG, anche questa è suddivisa per apparecchi, per facilitare la ricerca degli argomenti che interessano.
Alcuni BB offrono anche una sezione "Giochi", che consente di battersi (softwaricamente, s'intende) con altri utenti. Quasi tutti i BB visualizzano un messaggio introduttivo per gli utenti novelli, mentre maggiori informazioni sono contenute in appositi file da selezionare.

Caratteristiche di comunicazione HCAC	
Bulletin Board	
Numero di telefono (scambio dati)	vedere elenco a parte
Numero di telefono (informazioni)	non previsto
Costo	gratuito
Ore di funzionamento	vedere elenco a parte
Baud	300/300 (anche 1200/75)
Bit di dati	8
Bit di stop	1
Accesso a informazioni	in genere: <I>nfo
Termine dell'accesso	in genere: <G>oodbye
Posta elettronica	sì
Messaggi generici	sì
Software commerciale	no
Software gratuito	sì

ALCUNI BB OPERATIVI

È impossibile fornire un elenco completo dei BB esistenti, considerato anche il carattere spesso amatoriale che essi hanno.
Ecco però alcuni BB che, in Gran Bretagna, funzionano da tempo e sono ormai considerati quasi delle "istituzioni":

Bulletin board	Orario	Numero telefonico
TBBS Londra	24 ore	(GB) 01-348 9400
Mailbox-80 Liverpool	24 ore	(GB) 051-428 8924
The Blandford Board	24 ore	(GB) 0258-54494
CBBS Surrey	24 ore	(GB) 04826-25174

LOG ON: ACCESSO AL SISTEMA

Una volta collegato al BB, all'utente viene offerta gran parte delle informazioni occorrenti. Tuttavia, ecco alcuni utili suggerimenti.

- Controllare attentamente l'orario di accesso. I BB principali lavorano 24 ore su 24, ma altri solo poche ore al giorno (in genere nell'orario di ufficio). Ad alcuni sistemi si accede componendo il numero due volte: la prima semplicemente per far squillare il telefono e la seconda per effettuare il collegamento: si avverte così l'operatore che la chiamata non è diretta a lui, ma al modem.
- Assicurarsi che il proprio terminale (e anche il modem) sia pronto all'uso, prima della chiamata: alcuni BB "riattaccano" automaticamente se non intercettano entro 5 secondi il segnale della portante prodotta dal modem. In altri casi il collegamento può richiedere anche un paio di minuti, durante i quali il sistema riordina le informazioni, specialmente se un altro utente ha appena riattaccato.
- Essere preparati a dialogare con una voce umana. Anche i BB funzionanti 24 ore su 24 richiedono, talvolta, qualche intervento di manutenzione. In tal caso è l'operatore del sistema che risponde, informando sulla durata della momentanea interruzione nel servizio.
- Se il numero è occupato riprovare dopo qualche minuto: in genere la durata di una singola "conversazione" è limitata a una decina di minuti. In tal caso un modem dotato di chiamata automatica può essere molto comodo.
- A collegamento avvenuto, non si adoperino nomi fittizi (senza autorizzazione dell'operatore di sistema), né si lascino messaggi che potrebbero risultare offensivi per altri utenti (è una questione di educazione). Alcuni BB prevedono un controllo sull'etica dell'utente: se egli si comporta in modo inaccettabile, l'accesso al BB gli viene in seguito negato. Alcuni BB, al primo accesso, non prevedono la possibilità di scrivere nel sistema, ma solo di leggere e curiosare (con alcune restrizioni). Dal secondo accesso in poi, di solito, tali restrizioni vengono fatte cadere.

Esempio di collegamento ad un Bulletin Board

```
Welcome to the Blandford Board

Open 24 Hours Daily

CCITT at 300 Baud
8 Bits - No parity - 1 Stop

Member of A.F.P.A.S.

First Name? John
Last  Name? Smith
Calling from (Home Town)? London

TBBS Welcomes JOHN SMITH
Calling from London
Is this correct y

Terminal Width (10-132)? 52

     <A> TRS-100  <B> TRS-1/3/4  <C> APPLE
     <D> ATARI    <E> IBM-PC     <F> OSBORNE
     <G> BBC      <H> APRICOT    <I> COMMODORE 64

Enter letter of your terminal, or <CR>: <CR>

Can your terminal print lower case? y
Does your terminal need line feeds? y
How many Nulls (1-50)? 0

 The Blandford Board
 -------------------
 <N>ew user registration
 <B>ulletins
 <M>essaging section
 <U>tility area
 <I>nformation
 <C>hat with Sysop
 <G>oodbyte to terminate

Command: m

Messaging Section

-----------------
-----------------
<O>pen Message area
<M>ail - Private and Confidential
<S>IGs for Special Interests
<B>landford Computers (Os House)
<U>tility area
<E>xit messaging section

Command: o

Open Message Area

-----------------
<Q>uick-scan Open message headings
<R>ead Open messages
<L>eave an Open message
<M>ail - Private and Confidential
<E>xit Open message area
<G>oodbyte to terminate

Command: r

Type P to pause, S to stop, N to skip to next msg

<F>orward or <R>everse Multiple
<N>ew Messages
<M>arked Messages
<S>elective Retrieval
<I>ndividual Message(s)
<A>bort Retrieve

Which One? s

Select <F>rom, <T>o, or <S>ubject? s
Enter text to select by: Commodore

Reply to msg option (Y/N)? y

msg#:3330 *OPEN*
07/11/84 20:07:09 (Read 14 Times)
From: NORM PHILLIPS
To: ANY COMMODORE 64 USER
Subj: COMMODORE GEN: AS A NEW
USER OF THIS SYSTEM I WOULD BE
GRATEFUL FOR ANY HELP ANYONE
COULD OFFER ME TO GET THE BEST
FROM THE SYSTEM WITH MY
COMMODORE. I WOULD ALSO BE
INTERESTED IF THERE ARE ANY USERS
IN MY LOCAL AREA AND IF SO COULD
THEY CONTACT ME?

<N>ext msg, <R>eply, or <S>top? R

You are not authorised to write
messages on this board.

Msg. has replies, read now (Y/N)? n

At the start of each call you are set
for access to all combined boards.
You may toggle a board's activation
state by pressing the letter
indicated in the menu below. Any
boards not listed as active will not
display messages to you.

Active combined boards: Sales &
Wants, TRS-80, BBC,
APPLE, MODEMS, ATARI, C'64
<A> Sales & Wants <B> TRS-80
<C> BBC          <D> APPLE
<E> MODEMS       <F> ATARI

Which One?
```

Quando viene emesso il segnale della portante, battere un paio di <CR>: deve comparire il messaggio iniziale

I vari menu sono in genere abbastanza chiari e comprensibili. Se si è incerti, basta provare!

In molti BB il messaggio iniziale è chiaro e conciso, in altri viene visualizzata una sequela di codici: per evitarla spesso basta premere il tasto "S", passando direttamente all'immissione del proprio nominativo

Il BB chiede conferma dei dati immessi, controllando l'eventuale presenza di omonimi

OK

Vengono poi richieste le caratteristiche del terminale usato. A parte la larghezza in caratteri dello schermo, il menu prevede quasi tutti i tipi di terminale più diffusi

Se il proprio terminale non è elencato, basta premere <CR> per procedere all'immissione dei dati necessari

A questo punto viene chiesto di immettere una parola chiave (password) personale, da usare per i futuri collegamenti al BB. Questa serve soprattutto per impedire ad estranei la lettura di messaggi personali. Subito dopo (se non si tratta del primo accesso), il sistema comunica se vi sono messaggi destinati all'utente, in attesa di essere letti, altrimenti si passa direttamente al menu principale

Oltre al messaggio principale, si possono leggere anche le eventuali risposte fornite da altri utenti

Nella gran parte dei BB, prima di poter usufruire appieno dei servizi, occorre aver registrato il proprio nominativo. È abbastanza consueto che, al primo accesso, venga negata la possibilità di scrivere messaggi o di accedere a determinate sezioni del BB

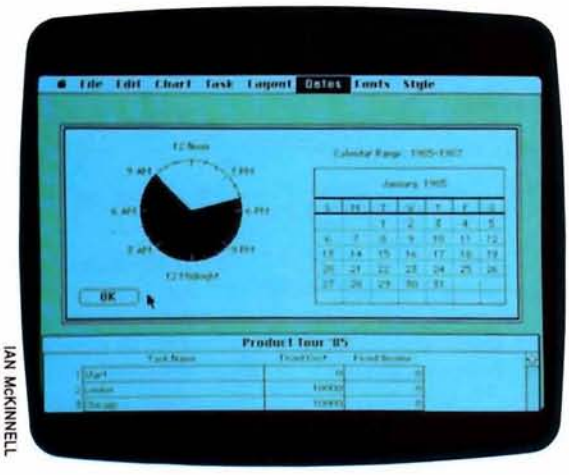
Ecco un tipico elenco dei gruppi con speciali interessi (SIG)

IAN McKINNELL sul MACINTOSH

Il percorso critico

Esaminiamo un package unico nel suo genere: il MacProject per il Macintosh

L'"analisi del percorso migliore", in inglese "Critical Path Analysis" o CPA è un potente strumento (in genere grafico) per l'organizzazione cronologica di attività o di eventi, in cui il progetto globale viene suddiviso in più passaggi o "percorsi", appunto, al fine di determinare la migliore successione temporale.
Si pensi, ad esempio, di dover progettare un piano di lavoro per la costruzione di una casa. Ogni particolare attività (la posa dei mattoni, l'imbiancatura, il montaggio dell'impianto elettrico e via dicendo) è in qualche modo connessa cronologicamente alle altre: l'intonacatura deve precedere l'imbiancatura ed entrambe vanno precedute dalla posa delle tegole del tetto. Saper organizzare con precisione l'intervento del falegname, del muratore e dell'elettricista significa ridurre drasticamente il tempo richiesto, globalmente, per realizzare l'impresa. Ogni intervento è caratterizzato da tre fattori: il momento in cui inizia, quello in cui finisce e la condizione di attuabilità (la necessità di aver portato a termine uno o più altri interventi, prima di partire con quello in questione).
La soluzione classica è quella di tracciare manualmente un grafico (una specie di cartina cronologica degli eventi), per organizzare la sequenza più corretta.



IAN McKINNELL

UN PROGETTO CRONOLOGICO

Il calendario incorporato nel "MacProject" (vedere illustrazione) consente di fissare, per ogni progetto, il numero di ore lavorative per ciascun giorno, tenendo anche conto di feste infrasettimanali e di altre condizioni particolari. Nella parte inferiore dello schermo, in questo caso, è visualizzata la tabella dei costi

Tuttavia, l'aggiornamento del grafico (per non parlare della sua stessa formulazione) implica uno sforzo notevole, con largo dispendio di tempo. Ecco perché questo genere di analisi viene quasi esclusivamente applicato in grandi progetti industriali.
Oggi esistono però degli ottimi programmi per computer, che rendono l'intera operazione molto semplice. Uno tra questi è il *MacProject*, appositamente realizzato per il Macintosh della Apple, che ben si presta a rappresentazioni grafiche di quantità numeriche.
In *MacProject*, una volta definite le caratteristiche del progetto, si ottiene un accurato "ruolino di marcia" assieme ad altre utili informazioni.
Se, in seguito, occorre apportare modifiche nelle scadenze, il programma aggiorna automaticamente il quadro generale del progetto, ricalcolando così nuovi tempi e relative scadenze.
Il manuale, chiaro e conciso, porta un esempio molto semplice per illustrare i principi di base: la preparazione di un "barbecue".
Questo "progetto" è suddiviso in cinque compiti distinti:

PREPARARE L'INSALATA
APPARECCHIARE LA TAVOLA
ACCENDERE IL BRACIERE
CUOCERE IL RISO
CUOCERE LE BISTECCHE

che conducono allo scopo finale del progetto:

MANGIARE LA CENA

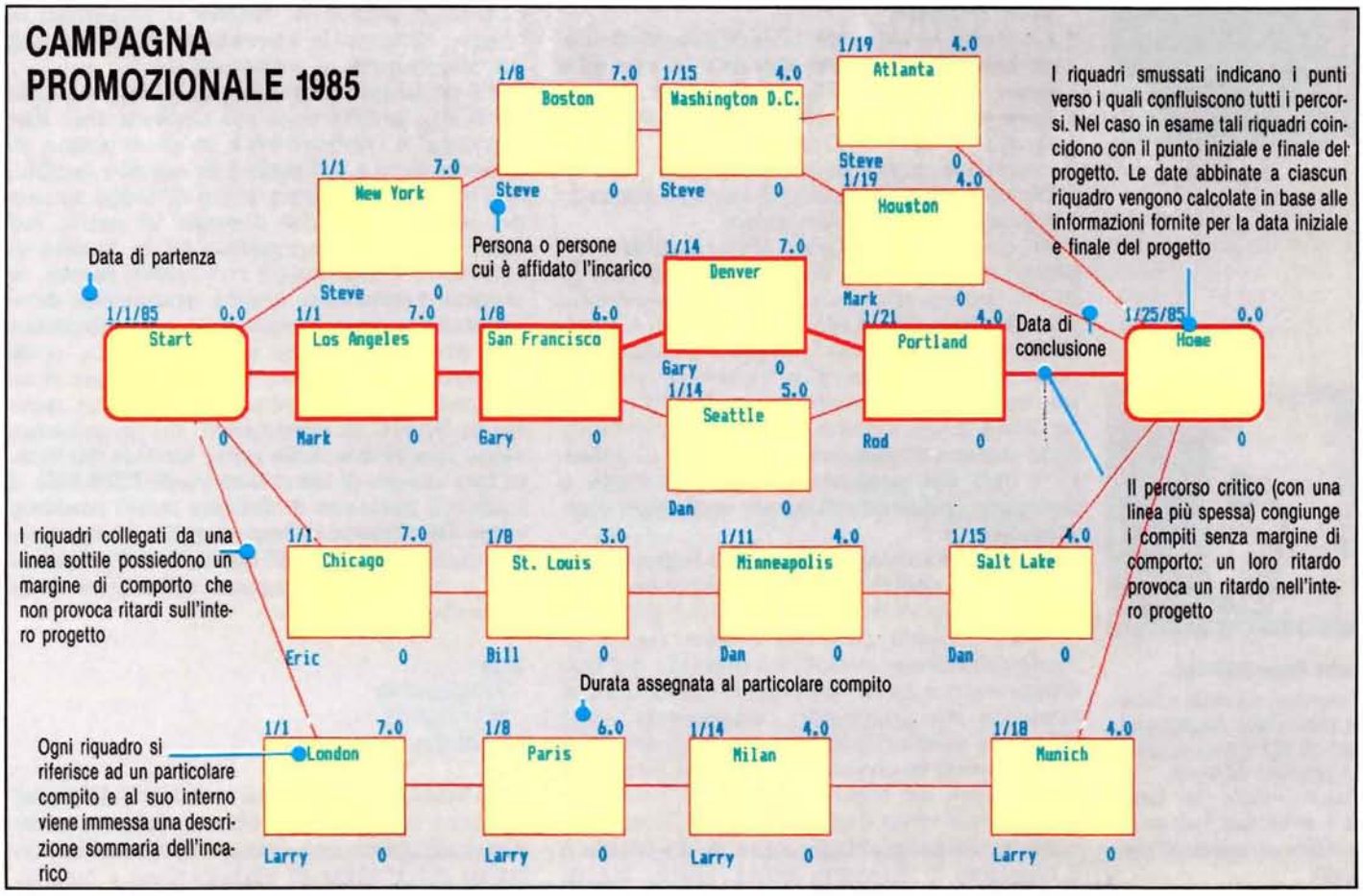
Alcuni dei compiti possono essere svolti simultaneamente, ma è impossibile cuocere le bistecche prima di aver acceso il braciere, cosicché il percorso critico va da "accendere il braciere" a "mangiare la cena" passando necessariamente da "cuocere le bistecche". Se c'è un ritardo nella preparazione della brace, l'intera cena subisce un ritardo. Analogamente, se la persona addetta all'insalata parte in ritardo, si corre il rischio che i commensali debbano affondare i denti in una bistecca ormai fredda.
Come menzionato dallo stesso manuale, non occorre certo un programma relativamente costoso come il *MacProject* per progettare una bisteccata fra amici, ma gli stessi principi di base si possono applicare alla costruzione di un pozzo petrolifero o all'organizzazione di un congresso.

I riquadri relativi ai vari compiti vengono creati sullo schermo tramite movimenti del Mouse. Le date iniziali e finali del progetto possono, ovviamente, essere modificate a piacimento. Ogni riquadro, all'interno del quale si immette una breve descrizione, può essere logicamente connesso ad altri, ancora una volta con semplici spostamenti del Mouse.
Anche le dimensioni e la posizione di ciascun riquadro possono essere modificate in base alle necessità.
Ciascun riquadro dispone di quattro zone in cui immettere informazioni di vario genere, abbinate al particolare compito, ad esempio: Risorse, Momento iniziale minimo, Momento finale minimo, Momento iniziale massimo e Momento finale massimo, Costi Fissi e Entrate Fisse. Al termine dell'elaborazione, il percorso critico attraverso i vari riquadri è evidenziato con un tratto più pronunciato, gli altri con un tratto molto più tenue.
Il programma, comunque, non si limita a questa analisi, ma prepara anche tutta una serie di diagrammi di enorme utilità. L'opzione RESOURCE TIMELINE, ad esempio, produce un grafico a barre che mostra la distribuzione delle risorse. L'opzione TASK TIMELINE è simile alla precedente, ma elenca sotto forma grafica i compiti, anziché le risorse (o i nomi delle persone incaricate).
Il periodo di margine (ossia la differenza, se esiste, tra il massimo intervallo concesso al particolare compito e il presunto lasso di tempo effettivamente necessario al suo compimento) viene evidenziato in grigio. Esso si riferisce al ritardo col quale un compito può iniziare, senza per questo interferire sui tempi globali del progetto.
L'opzione PROJECT TABLE produce una tavola di tutti i compiti, corredati da tutte le varie informazioni, evidenziando adeguatamente quelli "critici" (dai quali dipende l'intero progetto). Il programma può anche produrre tre distinte tavole per l'analisi dei costi dell'intero progetto.
MacProject è un'efficace implementazione dei principi e delle tecniche della CPA. Il manuale fornisce chiare spiegazioni (senza ricorrere a termini gergali), per cui si adatta anche a chi è completamente digiuno in materia. Questo programma può essere di aiuto nell'organizzazione di una vasta serie di progetti, particolarmente nell'ambito aziendale.

Esempio di progetto

Questa tavola prodotta dal "MacProject" rappresenta gli spostamenti di un gruppo di rappresentanti di una ditta, durante una campagna promozionale. La visualizzazione sotto forma di grafico facilita e velocizza la distribuzione degli incarichi e la loro collocazione temporale

MacProject: Per il Macintosh della Apple
Distributore: Apple Inc.
Autori: Debra Willett e Stephen D. Young della Solosoft
Formato: Su disco



Linguaggio disciplinato

Le origini storiche e i ferrei principi del PASCAL

Principali caratteristiche della programmazione in PASCAL

- Totale libertà di composizione e struttura dei listati di programma
- Flessibilità nella nomenclatura dei vari elementi nei programmi
- Possibilità di definire nuove "parole" di comando
- Sintassi semplice e coerente
- Elasticità nella definizione dei dati
- Struttura modulare di programmazione
- Controllo flessibile delle informazioni e delle procedure
- Ricursione naturale
- Ottimo tempo di compilazione e di diagnosi degli errori
- Un piccolo compilatore dalle grandi capacità

Il linguaggio PASCAL è stato concepito intorno al 1970 dal professor Niklaus Wirth di Zurigo. È così chiamato in onore del matematico e filosofo francese del 17° secolo Blaise Pascal, inventore della famosissima "Pascalina", la prima calcolatrice meccanica. La nascita del PASCAL è stata notevolmente influenzata dall'ALGOL 60, e costituisce la risposta di Wirth all'ALGOL 68, un linguaggio sovradimensionato e di grande complessità.

Secondo il suo creatore, il PASCAL doveva avere le seguenti caratteristiche:

- Permettere una più precisa definizione dei concetti e strutture di programmazione.
- Dimostrare che era possibile usare piccoli linguaggi, non legati ad un particolare hardware, ma dotati di un ricco e flessibile corredo di funzioni per la strutturazione delle informazioni e delle istruzioni, come strumenti per la soluzione di problemi di carattere generico.
- Aiutare a comprendere le tecniche necessarie per organizzare programmi di notevoli dimensioni e per gestire con sicurezza e affidabilità anche le più complesse applicazioni software.
- Ampliare le capacità di individuazione degli errori, soprattutto durante la compilazione, riducendo quindi gli errori di programmazione e introducendo un utile strumento per l'insegnamento della programmazione di un computer.
- Dimostrarsi un linguaggio sufficientemente adattabile ai microcomputer.

Tutti questi obiettivi sono stati raggiunti con pieno successo: un tipico compilatore PASCAL occupa appena 24 Kbyte ed è due volte più efficiente del FORTRAN. Il PASCAL possiede un "vocabolario" piuttosto limitato (appena 35 parole "chiave" o "riservate" rispetto alle cento e più di molti dialetti BASIC), ma è facile da usare e molto potente. Inoltre facilita la stesura di algoritmi e consente di definire i dati dei programmi con semplicità e coerenza, indipendentemente dalla loro complessità.

Il principale vantaggio di questo linguaggio è il suo modo chiaro e lineare di rappresentare anche gli algoritmi più sofisticati. I problemi di elaborazione possono essere risolti in modo più o meno semplice a seconda del tipo d'approccio e in questo senso il PASCAL si dimostra un linguaggio "amichevole", che non crea ulteriori problemi al programmatore ma concorre concretamente alla loro soluzione, oltre ad offrire molti altri vantaggi. Trattandosi di un linguaggio compilato, l'esecuzione dei programmi risulta molto rapida e il consumo di memoria molto ridotto. Infatti non è richiesta la simultanea presenza in memoria del listato sorgente e di un programma "interprete" del linguaggio, ma solo del codice "oggetto" (eseguibile) nella forma compilata.

Uno dei principali obiettivi del PASCAL è quello di proteggere il programmatore dai suoi stessi errori e di impedire l'esecuzione di programmi pieni di difetti. Questa particolarità può suscitare qualche perplessità iniziale, soprattutto in coloro che sono soliti programmare in BASIC, linguaggio più facile ma proprio per questa facilità favorisce la stesura di programmi contorti e, spesso, difficili da modificare o da rendere totalmente privi di errori.

In realtà è più facile programmare in PASCAL che non in BASIC, soprattutto quando si devono sviluppare programmi di grosse dimensioni. Il PASCAL è un linguaggio "disciplinato" e, per questo, consente un'eccellente individuazione e diagnosi degli errori: l'effettivo impegno occorrente per produrre programmi "a prova di bomba", in pratica, non va al di là dell'abituale attenzione necessaria a sviluppare un qualsiasi programma efficiente ed affidabile. Inoltre il risparmio di tempo, durante le operazioni di debugging del programma, è sorprendente.

Per il programmatore che passa dal BASIC al PASCAL, la differenza più evidente tra i due linguaggi è l'abbondanza, in quest'ultimo, di dichiarazioni e definizioni di aspetto insolito, che possono sembrare prive di scopo se non del tutto inutili. Ciò dipende in parte, dal fatto che il programmatore ha la facoltà di arricchire il linguaggio con nuove parole, di propria invenzione; questa operazione deve necessariamente precedere la parte dedicata alla programmazione vera e propria, onde consentire al PASCAL di comprendere il significato di tali nuove parole. Pertanto, mentre in BASIC la struttura di un programma viene specificata nella parte iniziale del listato (per mezzo di istruzioni quali GOSUB 5000), il PASCAL permette di definire nuovi comandi come PulisciSchermo o Pausa (seguito ad esempio dal numero di secondi) all'inizio del programma e di usarli in seguito all'interno della procedura principale:

```
begin
  PulisciSchermo;
  Write ('Salve!');
  Pause (3);
```

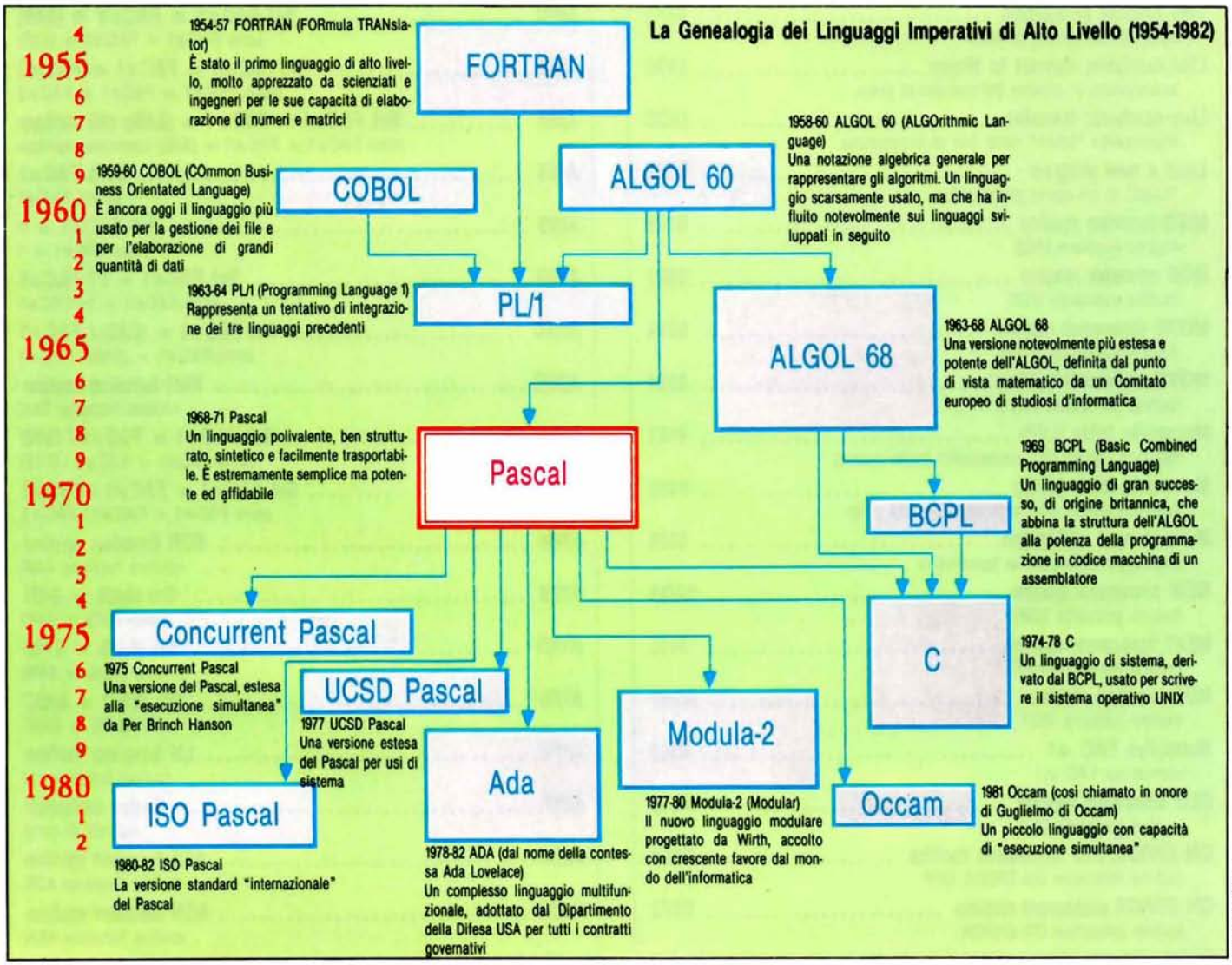
Il PASCAL favorisce una tecnica di programmazione metodica e molto concreta, presentando all'utente una visione altamente speculativa del sistema di elaborazione e consen-



Blaise Pascal (1623-1662)

Matematico, scienziato e filosofo, Blaise Pascal progettò e realizzò nel 1642 la prima calcolatrice meccanica del mondo. Il linguaggio PASCAL ha questo nome per ricordare il suo grande contributo alla scienza dell'informatica

tendogli di definire e rappresentare informazioni e procedure in forma logica e naturale. Tutto ciò rende i programmi molto affidabili e più facilmente trasportabili da un apparecchio all'altro, semplificando al tempo stesso le operazioni di debugging.
In generale, il PASCAL è senza dubbio il linguaggio che ha influenzato maggiormente i più recenti linguaggi di programmazione e lo stesso sviluppo del software applicativo.
Lo schema in basso mostra la genealogia dei linguaggi di alto livello evidenziando le principali influenze esercitate dai più importanti linguaggi "imperativi" compilati; sono pertanto stati omessi il LISP, il PROLOG e gli altri linguaggi di tipo "funzionale", come pure il FORTH, che è estraneo a qualsiasi classificazione. La corrente principale che ha influito sul PASCAL origina dall'ALGOL 60 e, come si può notare, non esiste linguaggio moderno che non abbia subito l'influsso, diretto o indiretto, del PASCAL. Una buona conoscenza di questo linguaggio favorisce quindi in modo significativo la comprensione dei linguaggi destinati ad affermarsi nei prossimi 10 anni: vale a dire il MODULA-2, l'OCCAM e l'ADA.
IL PASCAL è largamente adoperato per scopi didattici per le più svariate applicazioni gestionali e di sistema: dal software di carattere finanziario ai compilatori. Il "p-system", il famoso sistema operativo trasportabile, progettato alla fine del 1970 alla California University di San Diego, è stato sviluppato e scritto in PASCAL, come pure parte del software per Lisa e Macintosh della Apple (o nel suo derivato CLASCAL). Migliaia di programmatori professionisti hanno compiuto i loro primi passi in PASCAL, lavorando su elaboratori universitari DEC o VAX, ma anche sui più modesti APPLE II, che adottavano la versione del p-system nota come Apple PASCAL. Attualmente le principali case di software usano il PASCAL (o i linguaggi da esso derivati) per realizzare compilatori del linguaggio MODULA–2 o altro software applicativo e di sistema.
Ma la cosa che più sorprende del PASCAL è che si è universalmente affermato contando esclusivamente sulle proprie eccezionali qualità e non basandosi su interessi commerciali.
Più avanti viene descritto il linguaggio standard (ISO PASCAL), senza trascurare esempi relativi a specifiche macchine quando occorrono particolari chiamate per funzioni grafiche e sistema operativo.



DATABASE

Ecco la settima tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi

DESCRIZIONE	INDIRIZZO
LOAD command routine routine comando LOAD	**BF24**
LOCAL statement routine routine istruzione LOCAL	**9323**
LOG function routine routine funzione LOG	**ABA8**
LOMEM function routine routine funzione LOMEN	**AEFC**
LOMEM statement routine routine istruzione LOMEN	**926F**
Label definition in assembler definizione etichette (assembler)	**85A5**
Line number decoding decodifica numeri di linea	**97EB**
Line number encryption codifica dei numeri di linea	**88F3**
Line numbers; convert to binary conversione in binario del numero di linea	**889C**
Line numbers; tokenise inserimento "token" nelle line di programma	**88DB**
Load a new program "Load" di un nuovo programma	**BE62**
MID$ function routine routine funzione MID$	**B039**
MOD operator routine routine operatore MOD	**9E01**
MODE statement routine routine istruzione MODE	**939A**
MOVE statement routine routine istruzione MOVE	**93E4**
Mnemonic table (LSB) indirizzo tabella cod. mnemonici (parte bassa)	**8451**
Mnemonic table (MSB) indirizzo tabella cod. mnemonici (parte alta)	**848B**
Move to next statement passaggio all'istruzione successiva	**9880**
NEW command routine routine comando NEW	**8ADA**
NEXT statement routine routine istruzione NEXT	**B695**
NOT function routine routine funzione NOT	**ACD1**
Normalise FAC #1 normalizza FAC #1	**A303**
OLD command routine routine comando OLD	**8AB6**
ON ERROR OFF statement routine routine istruzione ON ERROR OFF	**B8E4**
ON ERROR statement routine routine istruzione ON ERROR	**B8F2**

INDIRIZZO	DESCRIZIONE
A4D0	**Set FAC#1 = FAC#1 — (&4B)** pone FAC#1 = FAC#1 — (&4B)
A4D6	**Exchange FAC#1 and (&4B)** scambio tra FAC#1 e (&4B)
A4DC	**Set FAC#1 = FAC#2** pone FAC#1 = FAC#2
A4E8	**Set MAN#1 = MAN#2** pone MAN#1 = MAN#2
A4FD	**Set FAC#1 = (&4B) — FAC#1** pone FAC#1 = (&4B) — FAC#1
A500	**Set FAC#1 = (&4B) + FAC#1** pone FAC#1 = (&4B) + FAC#1
A50B	**Set FAC#1 = FAC#1 + FAC#2** pone FAC#1 = FAC#1 + FAC#2
A606	**Set FAC#1 = FAC#1 * (&4B)** pone FAC#1 = FAC#1 * (&4B)
A613	**Set FAC#1 = FAC#1 * FAC#2** pone FAC#1 = FAC#1 * FAC#2
A656	**Set FAC#1 = FAC#1 * (&4B); chk ov/flow** pone FAC#1 = FAC#1 * (&4B) controlla overflow
A686	**Zero FAC#1** azzera FAC#1
A699	**Set FAC#1 = 1** pone FAC#1 = 1
A6A5	**Set FAC#1 = 1 / FAC#1** pone FAC#1 = 1 / FAC#1
A6AD	**Set FAC#1 = (&4B) / FAC#1** pone FAC#1 = (&4B) / FAC#1
A6BE	**TAN function routine** routine funzione TAN
A6E7	**Set FAC#1 = FAC#1 / (&4B** pone FAC#1 = FAC#1 / (&4B)
A6F1	**Set FAC#1 = FAC#1 / FAC#2** pone FAC#1 = FAC#1 / FAC#2
A7B4	**SQR function routine** routine funzione SQR
A7E9	**Set (&4B) = &471** pone (&4B) = &471
A7ED	**Set (&4B) = &746** pone (&4B) = &476
A7F5	**Set (&4B) = &46C** pone (&4B) = &46C
A7FE	**LN function routine** routine funzione LN
A897	**Series evaluator** calcolo di serie
A8D4	**ACS function routine** routine funzione ACS
A8DA	**ASN function routine** routine funzione ASN

Enterprise Sixty Four

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

L'home computer Enterprise Sixty Four, chiamato in un primo momento Elan e successivamente Flan, ha stabilito un record ancor prima di fare la sua comparsa sul mercato: sono infatti trascorsi ben due anni tra l'annuncio della sua uscita sul mercato e l'effettiva commercializzazione.
Come per il QL Sinclair, il ritardo è dipeso dalle difficoltà incontrate nel comprimere, in uno speciale chip "dedicato", un'impressionante quantità di funzioni.
Purtroppo però, nonostante l'impegno profuso, pare che non tutti i problemi siano stati risolti.
La struttura dell'apparecchio è indubbiamente originale, ma non così rivoluzionaria quanto facevano credere le prime indiscrezioni. La linea della custodia si discosta da quella "squadrata" tipica degli altri home computer: il suo frontale è curvo e smussato, mentre il colore grigio fumo della tastiera riprende quello della custodia.
La disposizione dei tasti, compresi quelli di Enter, Escape e Control è quella abituale. Inconsueta è invece la presenza di due tasti di cancellazione, Erase e Delete: cancellano, rispettivamente, il carattere a sinistra e quello a destra del cursore. Un terzo tasto, Hold, permette di "congelare", in qualsiasi momento, l'esecuzione di un programma.
I tasti di controllo, posti su entrambi i lati della tastiera, sono di colore verde. Sopra la tastiera alfanumerica sono collocati 8 tasti funzione blu. Inizialmente abbinati a 16 dei più usati comandi BASIC (LIST, AUTO e via dicendo), questi tasti sono riprogrammabili dall'utente.
Il tasto rosso Stop interrompe l'esecuzione di un programma che può essere riavviato premendo RUN o CONT. Sugli altri home computer questa funzione viene svolta dal tasto Escape che sull'Enterprise consente invece di passare da un livello di un programma a quello successivo.
I tasti sono leggermente più sporgenti rispetto alla tastiera del QL Sinclair, ma la loro forma, oltre alla "risposta", suscita qualche perplessità iniziale, che tuttavia si supera dopo un certo esercizio. Col tempo, infatti, i tasti si dimostrano più affidabili di quelli delle tastiere convenzionali.
Una particolarità del Sixty Four è il joystick incorporato, lungo 3 cm. e innestato in una membrana di gomma. Le sue dimensioni, seppur ridotte, consentono un perfetto controllo, anche nei giochi che richiedono rapidità e precisione di movimento.
Sul lato sinistro dell'apparecchio c'è una porta per cartucce, chiamata "ROM Bay". Il BASIC non è "residente", bensì viene fornito su cartuccia: probabilmente la Enterprise prevede di dotare in futuro il computer di altri linguaggi e programmi applicativi. Sebbene l'alloggiamento per le cartucce sia abbastanza profondo, il loro innesto risulta piuttosto difficoltoso. Sul lato opposto la macchina è dotata di un connettore "a pettine", probabilmente destinato ad espansioni di prossima produzione (tra le quali un'unità a dischetti).
Dal retro della macchina sporge il pulsante di Reset, che oltre a svolgere la normale funzione di azzeramento della memoria, ha anche un'altra funzione: premendolo due volte, infatti, si avvia il caricamento dei programmi su cartuccia. Questa seconda pressione è utile in quanto il sistema verifica la presenza della cartuccia solo all'accensione e non ai successivi reset.



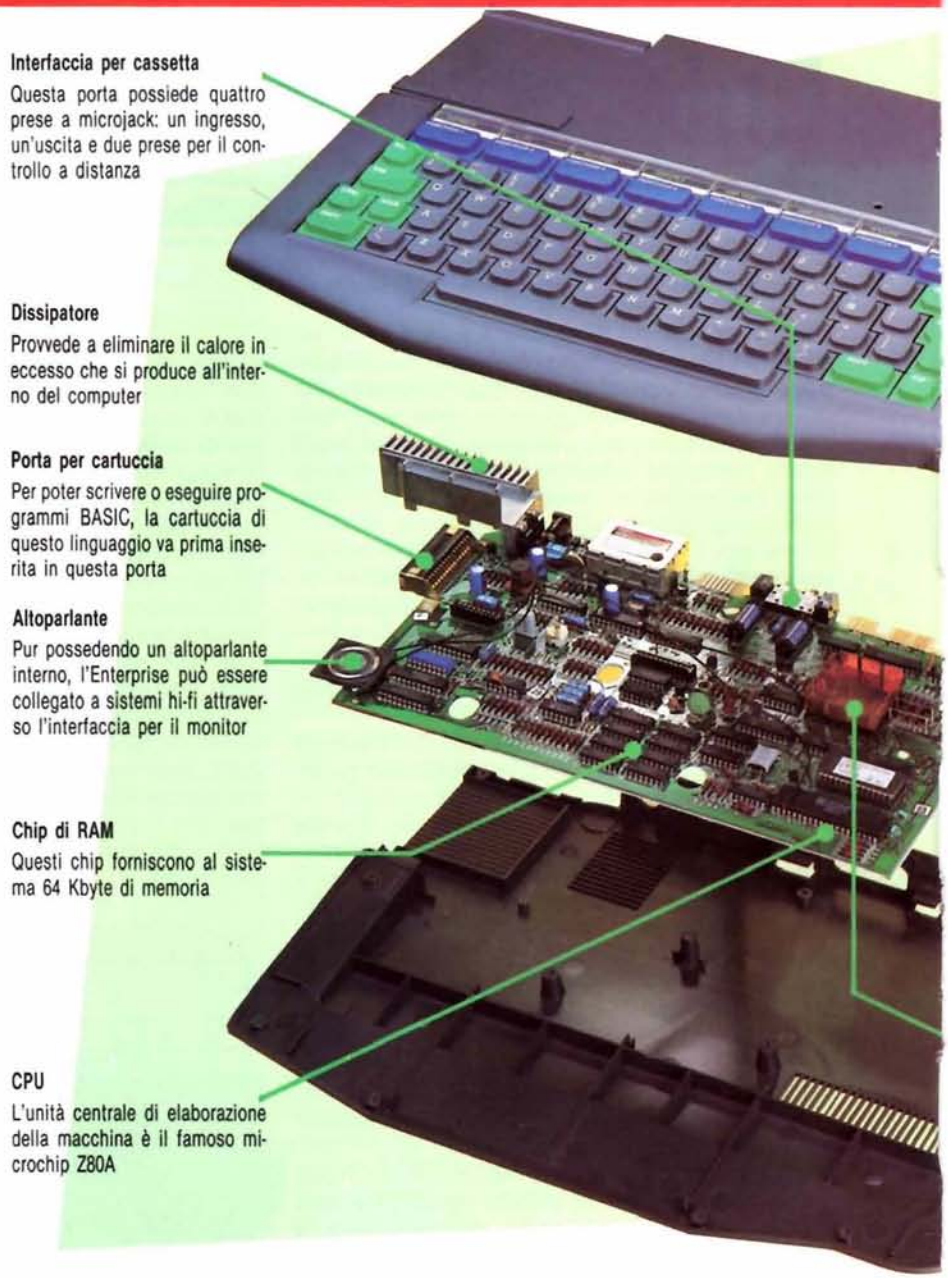
CHI TARDI ARRIVA...

L'Enterprise, il cui lancio era stato annunciato per il 1983, è in realtà apparso sul mercato 2 anni dopo. Questo ritardo ha indubbiamente influito negativamente sul successo dell'apparecchio: i nuovi micro prodotti in questo intervallo di tempo, infatti, hanno caratteristiche tecniche pari (in alcuni casi superiori) a quelle dell'Enterprise, che erano invece decisamente all'avanguardia nel 1983

Sempre sul retro della macchina c'è una serie di interfacce, tra le quali una coppia di porte per joystick, una porta per stampante parallela compatibile con lo standard Centronics e una porta seriale, compatibile con gli standard RS232/432, che permette all'Enterprise di collegarsi a sistemi "in rete" oppure a periferiche seriali e di comunicare con altri computer tramite un modem esterno. Infine, grazie a una presa per monitor, il computer può utilizzare un monitor a colori RGB.
Tutte queste interfacce adottano connettori "a pettine", che anche se da un lato consentono una maggior flessibilità d'impiego, dall'altro costringono gli utenti ad acquistare le apposite periferiche prodotte dalla stessa Enterprise.
Non è escluso che vengano in seguito prodotte interfacce di forma più consueta per collegare, ad esempio, joystick di tipo Atari.
Le altre porte inserite sul retro dell'apparecchio sono: un'interfaccia per cassetta con due prese per controllo a distanza, che permette il collegamento contemporaneo di due registratori a cassetta (uno in lettura e l'altro in scrittura); una porta RF per televisore e una presa d'alimentazione da collegare ad un trasformatore esterno.
All'accensione, sullo schermo compare il marchio di fabbrica dell'Enterprise. Se l'utente preme il tasto Enter, senza prima inserire la cartuccia del BASIC, il computer abilita l'uso del word processor interno. Questo programma ha caratteristiche superiori a quello del Commodore Plus 4 e offre una serie di opzioni per l'elaborazione dei testi, selezionabili mediante i tasti funzione. Sulla parte superiore dello schermo compare un elenco dei comandi più usati mentre, premendo il tasto f5, viene visualizzato un menu contenente gli altri comandi disponibili.
Una funzione particolarmente interessante è quella che permette di selezionare il modo testuale a 40 o a 80 colonne.
L'abbinamento tra lo schermo a 80 e il processore Z80 del computer indica che con buona probabilità, non appena saranno disponibili le unità a dischetti, il computer potrà adottare il CP/M.
Poiché l'apparecchio è privo di tasti per il cursore, i movimenti di questo vengono controllati con il joystick: questa originale soluzione può inizialmente provocare qualche difficoltà ma, con un po' di esercizio, i problemi di posizionamento del cursore tendono a scomparire.

Il BASIC dell'Enterprise

Se invece si accende il computer con la cartuccia BASIC inserita, è sufficiente premere il tasto Enter per entrare in BASIC. L'utente dispone di ben 49 Kbyte di memoria per i propri programmi. Grazie alle sue sofisticate funzioni, il dialetto BASIC (sviluppato dalla Intelligent Software) è indubbiamente uno dei principali punti di forza dell'apparecchio.
L'Enterprise possiede 4 generatori di suono



SOFTWARE RESIDENTE IN ROM





La presenza di un word processor incorporato, selezionato automaticamente se non è inserita alcuna cartuccia, differenzia l'Enterprise dalla maggior parte dei microcomputer. Il programma può essere usato nel modo a 40 o 80 colonne; la definizione dell'immagine sullo schermo è sorprendentemente buona



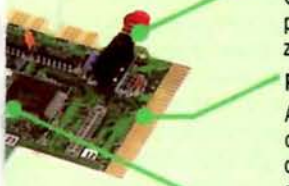


Un'originale caratteristica del Sixty Four è la presenza di una grafica "della tartaruga": esclusa l'istruzione PLOT, è abbastanza facile confondere i comandi di questa grafica con quelli del LOGO. È infatti possibile disegnare immagini complesse indicando alla "penna" la direzione e la distanza

IAN McKINNELL



Joystick
Al posto dei tasti per il cursore, l'Enterprise è dotato di un joystick incorporato, usato anche per la gestione dello "screen editor" dell'apparecchio



Pulsante di Reset
Questo pulsante riporta il computer alle condizioni iniziali, azzerando la memoria

Porta d'espansione
A questo bus parallelo verranno collegate le future espansioni dell'Enterprise, quali, ad esempio, un'unità a dischetti



Chip di suono
Questo chip, progettato appositamente per l'Enterprise e detto "Dave", controlla i quattro canali di suono e le 8 ottave disponibili sull'apparecchio

Chip per il video
Questo chip, fabbricato appositamente, dota il sistema di un'ottima grafica. È soprannominato "Nick", dal nome del suo progettista

che vengono selezionati tramite il comando SOURCE. Istruzioni come PITCH, DURATION e ENVELOPE consentono al programmatore di modificare l'emissione delle note. L'apparecchio è in grado di generare suoni stereofonici e l'utente può selezionare su quale altoparlante emettere una particolare nota, usando comandi LEFT e RIGHT.

I chip del Sixty Four producono suoni di elevata qualità, ma le caratteristiche dell'altoparlante interno sono piuttosto mediocri, appena superiori a quelle dello Spectrum Sinclair. Tuttavia è possibile collegare il computer ad un sistema hi-fi, attraverso la porta per il monitor.

L'Enterprise possiede ottime funzioni grafiche: sono infatti disponibili ben 8 modi grafici. L'utente può usare contemporaneamente 2, 4, 16 o 256 colori nei modi HIRES o LORES (rispettivamente, alta e bassa risoluzione). La risoluzione orizzontale di LORES è dimezzata rispetto a HIRES. Naturalmente il numero di colori visualizzabili dipende dalla risoluzione prescelta: HIRES 2, ad esempio, prevede due colori con una risoluzione di 640 x 180 pixel, mentre HIRES 256 offre ben 256 colori ma con una risoluzione di 80x180 pixel.

I colori vengono selezionati usando valori numerici compresi tra 0 e 255. Per gli 8 colori programmati all'interno del sistema è sufficiente digitare il nome (per esempio, RED o WHITE). Il comando RGB consente invece di selezionare e di mescolare assieme i colori. Ad esempio, sommando RGB (1,0,0) per il rosso e RGB (0,1,0) per il verde, si ottiene RGB (1,1,0), ossia il colore giallo.

Se l'Enterprise fosse stato commercializzato nel 1983, quando venne annunciato, l'elevata qualità del software interno gli avrebbe probabilmente consentito di sbaragliare la concorrenza.

Due anni di ritardo lo hanno invece penalizzato, ponendolo di fronte ad antagonisti molto più agguerriti, quali i nuovi modelli della Sinclair e della Commodore.

ENTERPRISE SIXTY-FOUR

DIMENSIONI

mm. 60 x 260 x 405

CPU

Z80A, funzionante a 4MHz

MEMORIA

64 Kbyte di RAM, 49 Kbyte dei quali disponibili per i programmi BASIC dell'utente e 48 Kbyte di ROM (32 Kbyte per il sistema operativo e il word processor e 16 Kbyte disponibili sulla cartuccia del BASIC)

VIDEO

Per il word processor si ha la scelta tra 40 o 80 colonne con 24 righe. I modi grafici sono otto. Quello con massima risoluzione offre 672 x 512 pixel. Il massimo numero di colori visualizzabili simultaneamente è 256

INTERFACCE

Interfaccia parallela compatibile con standard Centronics, due porte per joystick, porta seriale compatibile con la RS232, uscita per monitor RGB, uscita RF per TV e bus d'espansione

LINGUAGGI DISPONIBILI

BASIC

TASTIERA

Tipo QWERTY, con 69 tasti, di cui 8 per funzioni programmabili. Joystick incorporato

DOCUMENTAZIONE

Assieme al computer viene consegnato un manuale d'installazione, contenente utili diagrammi per le porte di connessione e un pratico manuale per la programmazione in BASIC

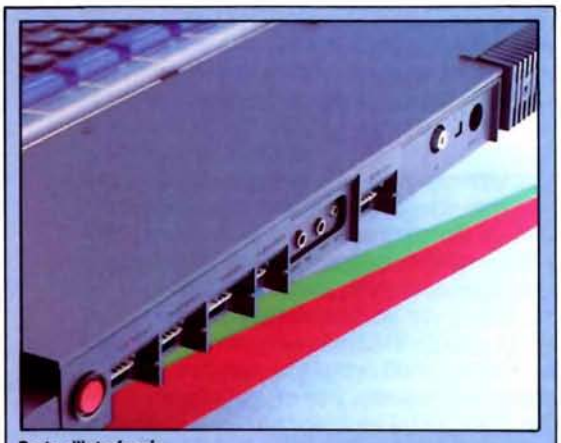
PRO

Questo computer possiede un'ottima versione di BASIC con eccellenti comandi grafici e audio (anche stereo). Word processor incorporato

CONTRO

Globalmente, questo apparecchio risente del ritardo intercorso tra l'annuncio e l'effettivo lancio commerciale. In generale, la parte hardware non pare essere all'altezza di quella software.

Il joystick incorporato non è più una novità e l'uso di interfacce non standard, almeno inizialmente, può scoraggiare i potenziali acquirenti



Porte d'interfaccia
La maggior parte delle porte d'interfaccia dell'Enterprise sono realizzate con connettori "a pettine", anziché con prese tipo DIN, più diffuse. Questa scelta consente di collegare l'apparecchio ad un maggior numero di periferiche, ma costringe ad acquistare appositi cavetti di connessione per l'utilizzazione di apparecchi con prese standard



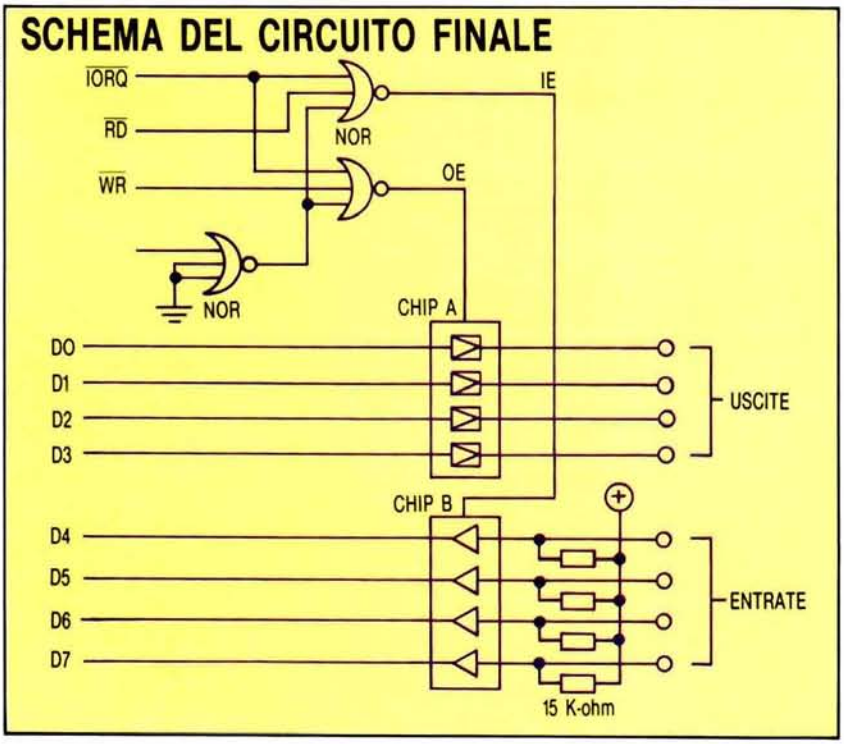
Cartuccia BASIC
Diversamente dalla tendenza più attuale, il BASIC sviluppato per l'Enterprise dalla Intelligent Software non è incorporato, bensì è contenuto in una cartuccia ROM da 16 Kbyte, da innestare in un alloggiamento sul lato sinistro della macchina. Questa scelta consente di sostituire al BASIC cartucce contenenti altri linguaggi di programmazione

CHRIS STEVENS

Interfaccia Spectrum

Dotiamo lo Spectrum di un'utilissima porta per applicazioni varie

KEVIN JONES

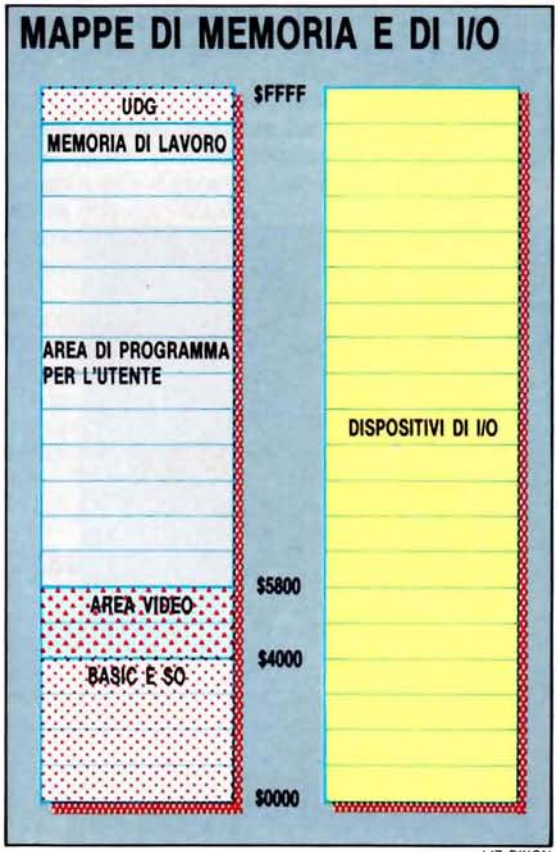


Linee logiche

Combinando le linee di richiesta di I/O, di lettura, di scrittura e d'indirizzo del bit 5 tramite una serie di porte NOR si possono produrre due segnali d'uscita. I segnali IE e OE abilitano le quattro linee d'ingresso e di uscita dei buffer (chip A e B), consentendo ai segnali di controllo generati dal computer di raggiungere i motori "passo passo" del robot e al software di controllo di ricevere e analizzare i segnali provenienti dai sensori del robot. Le linee d'ingresso e d'uscita sono attivate quando le rispettive linee $\overline{WR}$ e $\overline{RD}$ vengono poste ad 1

Mappe di memoria e di I/O

Il robot viene interfacciato con lo Spectrum attraverso una delle 65535 locazioni di questa mappa di I/O: le istruzioni IN e OUT, simili ai comandi PEEK e POKE per la mappa di memoria, consentono al programmatore di eseguire operazioni di lettura e scrittura sulla mappa di I/O



LIZ DIXON

Il Micro BBC e il Commodore 64 sono entrambi dotati di una porta utente multifunzionale che permette, ad esempio, di controllare il nostro veicolo robot. Lo Spectrum, al contrario, manca di una porta con simili caratteristiche. Anzi, se si esclude la porta per il registratore a cassette, l'apparecchio della Sinclair è totalmente privo di interfacce. Possiede invece, posteriormente, una porta d'espansione collegata a tutti i sistemi interni di connessione: al bus per i dati, al bus degli indirizzi, alle linee di controllo e ai diversi sistemi di scambio di segnali, compresa la linea di alimentazione. Queste linee possono dunque essere sfruttate per collegarlo a svariate interfacce, in quanto nulla vieta di costruire un'interfaccia che, innestata sullo Spectrum, svolga le funzioni di una porta utente.

Le porte utente dei computer impiegano un complicato chip, detto VIA (Versatile Interface Adaptor), capace di eseguire svariate operazioni. Il VIA fornisce al computer 8 linee di ingresso/uscita: ciascuna di esse (ossia ogni bit del byte di I/O) può essere posta, tramite software, in ingresso o in uscita. Inoltre possiede varie linee di controllo sui dati in ingresso e in uscita e un temporizzatore (clock) interno. L'interfaccia qui presentata è molto meno complessa, ma risponde perfettamente alle esigenze di controllo del veicolo robot, per il quale sono necessari 4 bit in ingresso e 4 in uscita. Poiché tale configurazione è "statica", non viene cioè modificata dal software, non occorre imitare del tutto la complessa configurazione del registro DDR/Dati del chip VIA.

Su quasi tutti i micro basati sul processore 6502, la porta utente e le altre interfacce vengono collegate alla circuiteria del computer in modo da essere "viste" dal processore come normali byte di memoria. Ciò significa che le interfacce divengono parte integrante della mappa di memoria del computer. Al contrario, lo Spectrum usa un processore Z80 dotato di un sofisticato sistema d'interfacciamento interno cosicché, invece di inglobare le unità di I/O nella stessa mappa di memoria della ROM e della RAM, lo Z80 ricorre ad una mappa di memoria separata ma "parallela" alla mappa principale. Per accedere ad essa vengono impiegati i comandi BASIC IN ed OUT, che corrispondono alle PEEK e POKE per la mappa principale.

Per distinguere la mappa di memoria da quella di I/O, lo Z80 si serve di un bit d'indirizzamento extra, che va ad aggiungersi ai 16 bit usati dal processore per indirizzare i 64 Kbyte di memoria. Questo bit corri-

	A10	A8	RFSH	M1	-12v	+12v	WAIT	-5v	WR	RD	IORQ	MREQ	HALT	NMI	INT	D4	D3	D5	D6	D2	D1	D0			D7	A13	A15
A11	A9	BUSACK	ROMCS	A4	A5	A6	A7	RESET	BUSRQ	U	V	Y	VIDEO	0v	IOROGE	A3	A2	A1	A0	CK	0V	0V		9V	5V	A12	A14

sponde alle due linee di controllo in uscita dal processore chiamate IORQ (Input/Output ReQuest) e MREQ (Memory REQuest). Quando il processore indirizza una locazione della memoria, la linea MREQ viene posta a 0, quando invece indirizza un'unità di I/O, viene posta a 0 la linea IORQ. Dato che la porta d'espansione dello Spectrum possiede sia le 16 linee d'indirizzo sia le due linee di controllo appena descritte, è possibile progettare un'interfaccia che risponda solo ad un particolare indirizzo e alla richiesta di I/O. Sullo Spectrum, alcune delle 65535 porte di I/O esistenti vengono adoperate per i collegamenti interni con la tastiera, con l'interfaccia per cassetta e con il generatore audio: di queste fanno parte le porte di I/O che si servono delle linee d'indirizzo da 0 a 4. L'interfaccia da realizzare usa unicamente la linea 5. Questa scelta implica la presenza dell'interfaccia in più aree della mappa di I/O (ossia con qualunque indirizzo che usi il bit 5), ma ciò consente di semplificare notevolmente il circuito.

Poiché l'interfaccia gestisce l'ingresso e l'uscita dei dati, è necessario usare sia le linee di scrittura (WR) e di lettura (RD) sia quelle degli indirizzi e di controllo dello Z80. L'interfaccia, quindi, risponde sia all'azzeramento delle linee IORQ e A5, sia all'azzeramento di una delle linee WR o RD, collegando il bus dati del computer alle linee per i dati del veicolo robot. Il riquadro sulla destra mostra la tavola della verità per l'interfaccia.

I segnali IE e OE abilitano rispettivamente le unità d'ingresso e d'uscita dell'interfaccia. Va detto che non può mai verificarsi uno "stato" finale in cui tutte le linee (comprese RD e WR) si trovino contemporaneamente a 0 (infatti il microprocessore non può leggere e, nello stesso momento, anche scrivere in una stessa locazione).

Quando le linee IORQ, RD e A5 vengono poste a 0 (ad esempio, se il processore legge il contenuto della porta di I/O 36), la linea IE dell'interfaccia viene posta a 1 e le quattro linee d'ingresso vengono collegate ai quattro bit alti del bus dati del computer, passando attraverso il chip "buffer" A. Quando vengono poste a zero le linee IORQ, A5 e WR, la linea OE passa a 1 e i quattro bit bassi del bus dei dati vengono collegati alle quattro linee d'uscita dell'interfaccia. Infine, quando OE viene posto a 0 (perché il processore si rivolge a un altro indirizzo), il chip B mantiene la configurazione in uscita, finché il processore non torna a indirizzarlo: questo procedimento è detto "latching, ossia "bloccaggio" dei dati. Per qualunque altra configurazione delle quattro linee, i collegamenti dell'interfaccia vengono tenuti isolati dal bus dei dati e non influiscono in alcun modo sul computer o sull'interfaccia stessa.

Così concepito, il progetto del circuito dell'interfaccia per il controllo del robot si basa su tre chip: quello contenente le tre porte OR a ingresso triplo (una quarta porta rimane inutilizzata), il buffer quadruplo (chip A) e il chip di bloccaggio quadruplo (chip B).

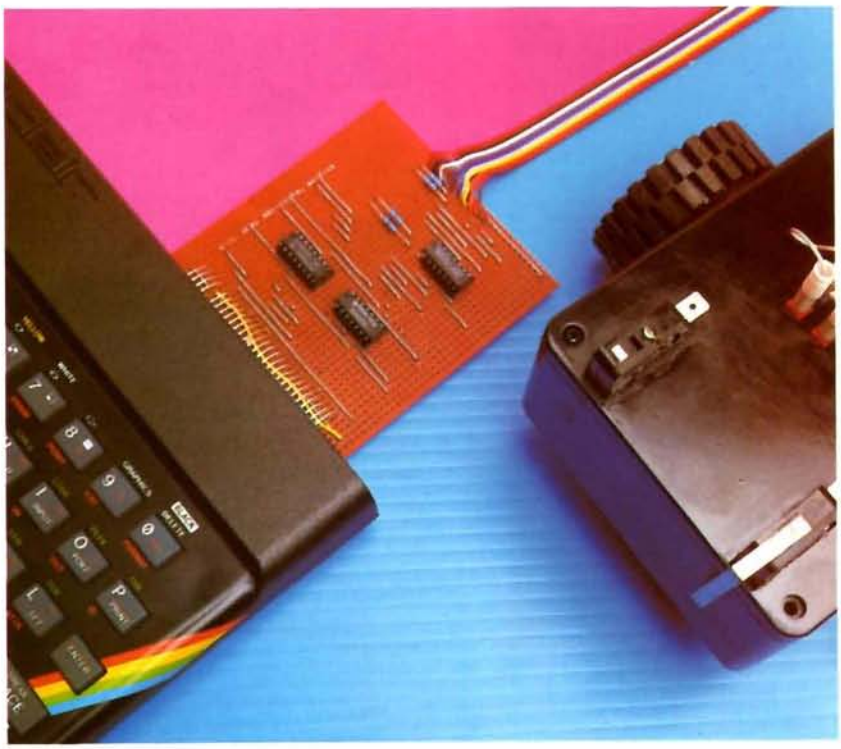
Tavola della verità per l'interfaccia

IORG	WR	RD	A5	IE	OE
0	0	0	0	—	—
0	0	0	1	—	—
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	—	—
1	0	0	1	—	—
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

La tavola della verità mostra i segnali generati sulle linee di abilitazione dell'ingresso e dell'uscita, in base a tutte le possibili combinazioni dei 4 segnali in ingresso. L'azzeramento contemporaneo di WR e RD in pratica non si verifica mai, poiché non è possibile eseguire un'operazione simultanea di lettura e di scrittura. Azzerando IORQ, RD e A5 si abilita l'ingresso dei dati e collegando queste tre linee a una porta NOR si ottiene il segnale IE. Azzerando invece le linee IORQ, WR e A5 si abilita l'uscita dei dati e collegando queste linee ad una seconda porta NOR si ottiene il segnale OE

Interfaccia in funzione

Il Micro BBC e il Commodore 64 possiedono speciali chip di I/O, che permettono un collegamento diretto con il robot. Lo Spectrum necessita invece di un'interfaccia appositamente progettata da innestare sulla porta d'espansione. La sua presenza permette al software dello Spectrum di controllare il robot



LIZ HEANEY

La gestione dei file sul BBC

Esaminiamo la chiamata OSFILE del sistema operativo del BBC

Il sistema di archiviazione di un computer gestisce lo scambio di informazioni tra la memoria del computer e le memorie di massa esterne, quali le cassette o i dischetti. Il Micro BBC può usare entrambe le unità ma possiede anche altri sistemi di archiviazione, come il sistema "in rete" Econet, le cartucce ROM e il telesoftware. Contrariamente a quanto ci si potrebbe aspettare, il sistema di archiviazione del Micro BBC adopera le stesse routine del SO per tutti i sistemi di cui dispone, ma ciò non significa che i diversi sistemi funzionino allo stesso modo. Infatti, per organizzare la struttura fisica del file di ciascun sistema (ossia la disposizione dei byte di dati all'interno dell'unità di immagazzinamento), l'apparecchio deve possedere apposite ROM per ciascun sistema di archiviazione, ma queste vengono interfacciate con la struttura logica del file (record, campi, ecc.) tramite le routine standard del SO.

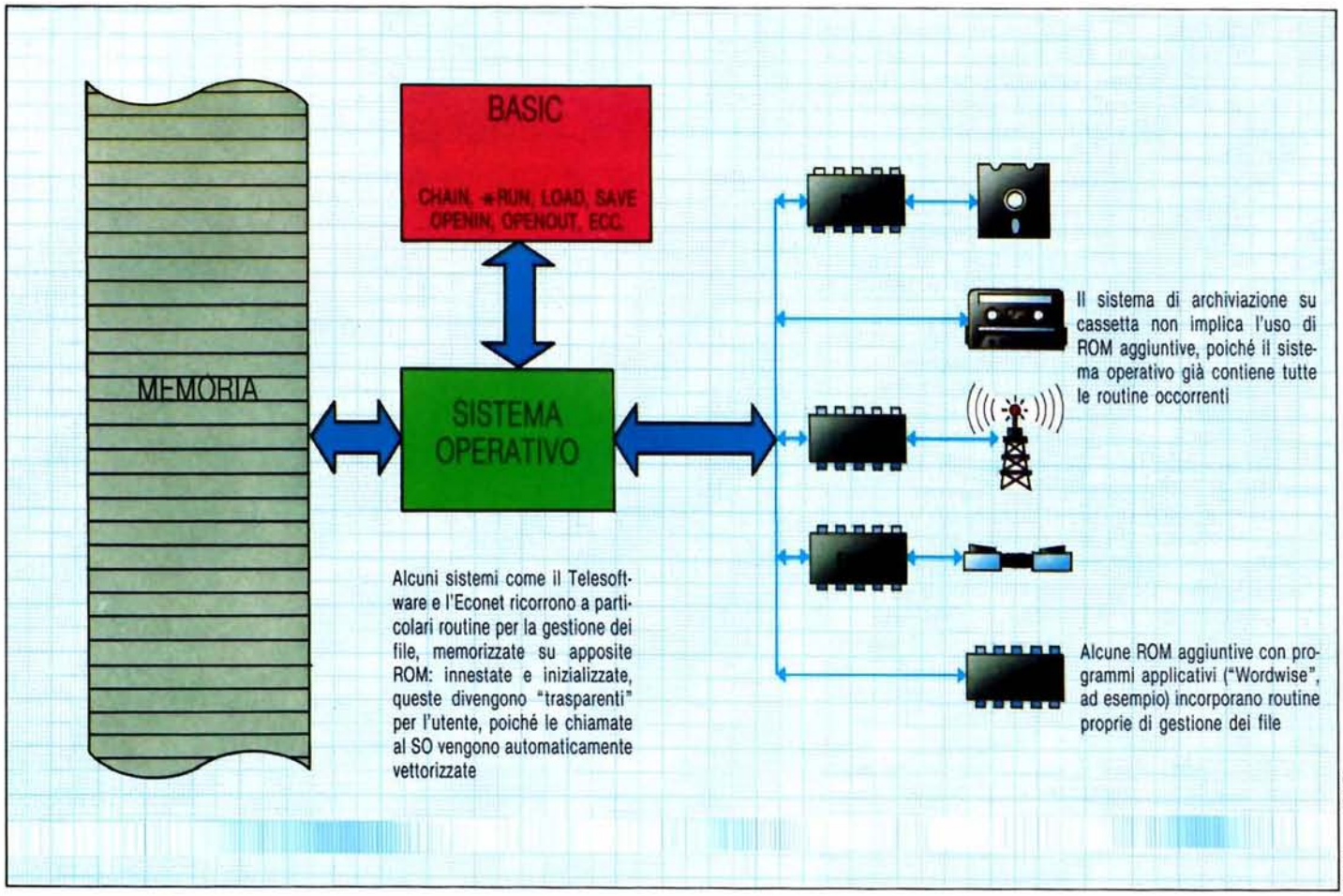
Le chiamate ROM del sistema di archiviazione adottato sul Micro BBC sono vettorizzate, cosicché quando l'utente seleziona un particolare sistema di archiviazione, al tempo stesso ordina al SO di conformarsi ad esso. Ne risulta che un programma scritto usando esclusivamente routine "autorizzate" del SO, può lavorare assieme a qualsiasi tipo di memoria esterna. Ovviamente, alcuni sistemi di archiviazione non consentono particolari operazioni. Ad esempio, non ha senso tentare di usare un'istruzione SAVE su di un chip di ROM.

Un sistema di archiviazione può essere abilitato in molti modi: vediamo alcune delle più semplici tecniche di selezione.

• Per poter attivare un particolare sistema di

Archivi & Archivi

Il sistema di archiviazione offre una serie di routine per lo scambio di informazioni tra la memoria del computer, le memorie di massa e altri sistemi di immagazzinamento (compresi il Telesoftware e i sistemi "in rete)



archiviazione con la semplice pressione del tasto Reset, occorre innestare la relativa ROM, all'interno della macchina, in una speciale posizione, detta di "default". Di solito, quando il computer possiede più ROM dedicate al sistema di archiviazione, viene selezionata quella posta immediatamente a sinistra della ROM BASIC, sempre che quest'ultima sia stata inserita sul primo zoccolo di destra delle ROM impaginate.

- Una seconda tecnica prevede l'uso abbinato del BREAK e di un altro tasto. Ad esempio, premendo contemporaneamente BREAK ed N in presenza della appropriata ROM all'interno della macchina, viene abilitato il sistema di archiviazione Econet.
- Tramite comandi *. Il comando *TAPE 12 seleziona il sistema di archiviazione su cassetta con velocità di trasmissione di 1200 baud, *DISC abilita l'archiviazione su dischetto, *TELESOFT il sistema Telesoftware e *NET il sistema Econet. Naturalmente anche gli ultimi tre comandi richiedono la presenza delle rispettive ROM all'interno del computer.

In pratica, il sistema di archiviazione viene usato ogniqualvolta l'apparecchio viene acceso, anche se solo per caricare (LOAD) o salvare (SAVE) un programma. Entrambi questi comandi impiegano routine del sistema di archiviazione proprie del SO: la stessa cosa vale per i comandi CHAIN, OPENUP, OPENOUT, OPENIN, INPUT, PRINT e per i comandi * mostrati nella tabella a pagina seguente. Esistono tuttavia altri comandi * per l'archiviazione, forniti su apposite ROM.

Prima di passare all'esame delle numerose routine ROM coinvolte nelle operazioni di archiviazione del Micro BBC, va detto che la descrizione si limita ad alcuni dei possibili e principali impieghi delle diverse chiamate.

La chiamata OSFILE

La chiamata OSFILE opera su interi file di dati e ad essa ricorrono comandi SAVE, LOAD, *SAVE ed altri che operano su grosse quantità di dati. A tali file viene associata una serie di informazioni, chiamate "attributi, riguardanti l'indirizzo di caricamento, quello d'esecuzione, la lunghezza del file e così via. OSFILE consente al programmatore di modificare tali attributi adattandoli al particolare sistema di archiviazione in uso (all'unità a dischetti, per esempio). Come già visto per la chiamata OSWORD, anche OSFILE utilizza un blocco dei parametri, il cui indirizzo viene comunicato mediante i registri X e Y. Inoltre OSFILE è vettorizzata e viene chiamata all'indirizzo &FFDD. L'operazione eseguita dipende dal valore contenuto nel registro A, al momento della chiamata.

Le due tabelle a fianco illustrano la struttura del suo blocco dei parametri e le operazioni eseguite.

Poiché su un Micro BBC standard è sufficiente un indirizzo a 16 bit per indicare le locazioni di memoria, l'apparecchio prende in considerazione solo il byte meno significativo e quello adiacente dei numeri a quattro byte che specificano gli indirizzi.

Esaminando la tabella delle operazioni eseguite si nota facilmente che il sistema di archiviazione su cassetta prevede un numero di

Blocco dei parametri per la chiamata OSFILE	
Offset dall'indirizzo nei registri X ed Y	**Descrizione**
0	BMS dell'indirizzo del nome di file
1	BPS dell'indirizzo del nome di file
2	BMS dell'indirizzo di caricamento del file
3	Non usato
4	Non usato
5	BPS dell'indirizzo di caricamento del file
6	BMS dell'indirizzo di esecuzione del file
7	Non usato
8	Non usato
9	BPS dell'indirizzo di esecuzione del file
10	BMS dell'indirizzo iniziale dei dati da scrivere nel file per l'operazione di scrittura, oppure indirizzo di destinazione per una lettura
11	Non usato
12	Non usato
13	BPS dell'indirizzo iniziale dei dati
14	BMS di (indirizzo finale dei dati + 1) per un'operazione di scrittura oppure informazioni sugli attributi del file per una lettura
15	Non usato
16	Non usato
17	BPS dell'indirizzo iniziale dei dati

BMS = Byte Meno Significativo, BPS = Byte Più Significativo

Operazioni eseguite dalle chiamate OSFILE		
Contenuto di A	**Descrizione dell'Operazione eseguita**	
	Per disco	**Per nastro**
0	Salva l'area di memoria	Salva l'area di memoria
1	Scrive sul catalogo	Carica il file
2	Scrive l'indirizzo di caricamento	Carica il file
3	Scrive l'indirizzo di esecuzione	Carica il file
4	Scrive gli attributi del file	Carica il file
5	Legge il catalogo	Carica il file
6	Cancella il file	Carica il file
255	Carica il file	Carica il file

operazioni minore rispetto a quello su dischetto. Le informazioni sugli attributi del file dipendono dal sistema di archiviazione impiegato e ciò è particolarmente utile per le operazioni su dischetto, in quanto è possibile, ad esempio, cambiare l'indirizzo di caricamento di un file semplicemente modificando il relativo valore nel blocco dei parametri ed eseguendo la chiamata OSFILE dopo aver inserito l'appropriato valore nel registro A.
Per meglio chiarire il funzionamento di OSFILE, consideriamo il seguente programma, che si "auto-salva" tramite una chiamata OSFILE:

```
10 DIM blocco 20
20 DIM nome 10
30 BASIC = &8023: REM per il BASIC II, &801F per il BASIC I
40 X%=blocco MOD 256: REM punta al blocco dei parametri
50 Y%=blocco DIV 256
60 A%=0: REM indica un'operazione di scrittura
70 blocco!0=nome: REM indirizzo del nome del file
80 blocco!2=PAGE: REM l'indirizzo di caricamento del
90                 : REM programma è qui PAGE
100 blocco!6=BASIC: REM l'indirizzo d'esecuzione è qui
110                 : REM l'ingresso del BASIC
120 blocco!10=PAGE: REM il primo byte da scrivere
130                 : REM è qui PAGE
140 blocco!14=(TOP+1) :REM fine dei dati+1
150 $nome="PIPPO": REM usare un nome di file consentito
160 CALL &FFDD :REM esecuzione
170 END
```

Per cancellare un file dal dischetto con una chiamata OSFILE basta depositare in memoria il nome del file, puntare su di esso i primi due elementi del blocco dei parametri, assegnare il valore 6 al registro A e, infine, chiamare OSFILE. Dopo una chiamata di OSFILE con A=5, i byte 2 e 5 del blocco dei parametri contengono l'indirizzo di caricamento del file, mentre i byte 6 e 9 contengono l'indirizzo d'esecuzione. Se il nome del file trovato all'indirizzo indicato dal blocco dei parametri non è presente sul dischetto, al ritorno dalla chiamata A contiene il valore 200, mentre se il file viene trovato, A contiene il valore 1.

Comandi * che usano le routine del sistema di archiviazione

Comando *	Descrizione
*CAT oppure *.	Visualizza un catalogo dei file a disposizione sul sistema di archiviazione
*SAVE	Salva un blocco di memoria RAM in un file. Ad esempio: *SAVE "nome di file" inizio fine esecuzione dove "inizio" rappresenta l'indirizzo esadecimale del primo byte da salvare, "fine" quello dell'ultimo byte da salvare e "esecuzione" l'indirizzo di esecuzione del file salvato. Il parametro "esecuzione" è facoltativo: se non specificato, l'indirizzo di esecuzione assume il valore di "inizio"; "fine" può essere sostituito da "+ lunghezza", assegnando a "lunghezza" il numero di byte da salvare
*LOAD	Carica nella memoria del computer un file, a partire da un indirizzo dato. Il comando ha il seguente formato: *LOAD "nome di file" indirizzo dove "indirizzo" indica l'indirizzo esadecimale iniziale per il caricamento dei byte. In sua assenza, i byte vengono caricati a partire dall'indirizzo posseduto dal codice al momento del salvataggio
*RUN	È simile ad una versione in codice macchina del comando CHAIN del BASIC: *RUN "nome di file" carica il file a partire dall'indirizzo dal quale i byte erano stati originariamente salvati ed esegue il codice come se fosse un programma in codice macchina, iniziando dall'indirizzo di esecuzione salvato in precedenza
*SPOOL e *EXEC	Sono già stati descritti a pag. 1360
*OPT 1,x	Comunica al SO se i messaggi devono essere visualizzati durante le operazioni di archiviazione dei file: x=0 nessun messaggio; x=1 messaggi parziali, x=2 messaggi completi
*OPT 2,x	Stabilisce il tipo di risposta del SO, quando si verifica un errore durante le fasi di archiviazione dei file. Se x=0 gli errori vengono ignorati; x=1 invita l'utente a ripetere l'operazione al verificarsi di un errore; x=3 interrompe l'attività del sistema di archiviazione alla comparsa di un errore
*OPT 3,x	Indica quale deve essere il tempo di intervallo tra due registrazioni consecutive su nastro, eseguite dal sistema di archiviazione su nastro. x esprime il tempo di ritardo in decimi di secondo
*OPT 4,x	Viene usato solo con i sistemi di archiviazione su dischetto ed è stato incluso in questa tabella per la sua grande utilità. Seleziona le opzioni SHIFT—BREAK su disco. Per maggiori dettagli consultare il manuale del Disk Filing System

La Foresta Incantata

Ecco qualche ultima considerazione, assieme al listato finale di questo gioco

Foresta Incantata è un gioco breve, sviluppato appositamente per mettere in luce alcune tecniche di base della programmazione dei giochi di tipo adventure. Per motivi di spazio sia la trama del racconto, sia la quantità delle opzioni a disposizione sono relativamente limitate. Comunque, la struttura del programma mostra come sia possibile sviluppare un gioco d'avventura attorno ad una semplice intelaiatura, articolata su una mappa che illustra le diverse locazioni e le loro reciproche interazioni durante lo svolgimento dell'azione. Indipendentemente dal numero delle locazioni rappresentate, la formulazione del gioco parte sempre da una mappa, in cui sono evidenziati anche gli oggetti in gioco e i pericoli "speciali". Ovviamente, i dettagli sia della trama del gioco sia dei pericoli principali devono essere progettati prima di passare all'effettiva codifica. Per semplificare il lavoro e stabilire con precisione le regole del gioco, spesso conviene disegnare la mappa su un foglio di carta quadrettato.

Solo quando tutto è chiaro si può dare inizio alla fase di programmazione.

Di solito per prima viene codificata l'immagine bi-dimensionale della mappa all'interno di una matrice. Nel gioco *Foresta Incantata*, le informazioni per le diverse locazioni sono registrate in matrici bidimensionali contenenti la descrizione della locazione e la lista in codice delle uscite possibili. La stessa tecnica viene seguita per gli oggetti in gioco: gli elementi della loro matrice bidimensionale contengono la descrizione dell'oggetto e la locazione in cui si trova. Si ottiene così una semplice banca dati che rappresenta lo scenario di gioco.

Si passa poi alla programmazione di una serie di routine di controllo, che permettono al giocatore di muoversi all'interno delle locazioni e di raccogliere o posare oggetti. In questa fase manca ancora una precisa definizione delle regole del gioco, per quanto siano già presenti gli oggetti che consentiranno al giocatore di affrontare e superare i pericoli "nascosti" nelle diverse locazioni speciali.

Questa versione del programma è stata scritta per il Commodore 64, ma funziona sulla maggior parte degli apparecchi dotati di dialetti BASIC tipo Microsoft. In alcuni casi, tuttavia, occorre modificare leggermente le procedure per la generazione casuale dei numeri e la cancellazione dello schermo. Il riquadro "Basic & Basic" a fianco mostra le correzioni necessarie per ottenere una versione per il Micro BBC. Poiché lo Spectrum adotta una propria gestione delle matrici stringa, le modifiche richieste sono molto numerose: conviene quindi fare riferimento ai listati forniti in precedenza. Tuttavia, per facilitare l'adattamento del listato allo Spectrum, possiamo consultare la seguente tabella di conversione che riassume tutti i nomi delle variabili stringa usati nel programma.

Un gioco d'avventura si basa essenzialmente sulla capacità del giocatore di individuare le procedure nascoste e sulla sua abilità nel superare i pericoli che incontra: l'approfondita analisi della struttura di *Foresta Incantata* ha purtroppo svelato il meccanismo del gioco e ne ha "smascherato" il funzionamento interno. *Digitaya*, al contrario, permette anche allo stesso programmatore di cimentarsi in un gioco d'avventura senza conoscerne la soluzione vincente.

Tabella di conversione per variabili stringa Spectrum

Versione Microsoft	Versione Spectrum	Funzione delle variabili
LN$( )	L$( )	Descrizione della locazione
EX$( )	E$( )	Uscite
IC$( )	I$( )	Oggetti in possesso del giocatore
IV$(,)	V$(,)	Oggetti in gioco
SN$	S$	Contiene la frase da formattare
OW$	O$	"Vecchia parola" nella routine di formattazione
NW$	N$	"Nuova parola" nella routine di formattazione
EX$	X$	Uscite dalla locazione corrente
IS$	I$	Istruzione corrente
DR$	D$	Direzione indicata dall'istruzione
NN$	R$	Parte non verbale dell'istruzione
VB$	B$	Parte verbale dell'istruzione
CD$	C$	Parola di codice

Basic & Basic

Micro BBC:

Sostituire le seguenti linee nel listato "Foresta Incantata":

```
207 RND (—TIME)
210 P = RND (10)

1160 CLS

4190 REPEAT: A$ = GET$: UNTIL A$ = "S" OR A$ = "N"
4535 CR = RND(3)

6067 CC = 0
```

IL LISTATO FINALE

```
130 **  FORESTA  **
140 ** INCANTATA **
180 :
200 GOSUB 6000: REM LETTURA DATI NELLA MATRICE
205 GOSUB 1000: REM PROLOGO DEL RACCONTO
207 R=RND (-1)
210 P=INT(RND(TI)*10+1): REM PUNTO INIZIALE
220 :
230 REM ** INIZIO DEL CICLO PRINCIPALE **
240 MF=0: INIZIALIZZA IL FLAG
245 PRINT
250 GOSUB 2000: REM DESCRIVE UN AMBIENTE .
255 GOSUB 2300: REM INDICA LE USCITE
257 GOSUB 2700: REM P E' UNA LOCAZIONE SPECIALE?
258 IF SF=1 THEN 300: REM COMANDO SUCCESSIVO
260 PRINT:INPUT "COMANDO";IS$
270 NN$=IS$: GOSUB 3500: REM SEPARA I COMANDI
275 IF F=0 THEN 260: REM COMANDO NON VALIDO
280 GOSUB 3000: REM COMANDI NORMALI
290 IF MF=0 THEN PRINT:PRINT "NON CAPISCO"
300 IF MF=1 THEN 240: REM NUOVA LOCAZIONE
310 IF MF=0 THEN 260: NUOVO COMANDO
320 :
990 END
1000 REM ** S/R PROLOGO DEL RACCONTO **
1010 SN$="BENVENUTO NELLA FORESTA INCANTATA"
1020 GOSUB 5500: REM FORMATTAZIONE
1030 PRINT
1040 SN$="TI SEI APPENA SVEGLIATO DA UN PROFONDO SONNO E "
1050 SN$=SN$+"TI TROVI IN UNA GRANDE FORESTA. IL SUOLO E' "
1060 SN$=SN$+"MORBIDO E ASCIUTTO. NON RICORDI COME SEI GIUNTO "
1070 SN$=SN$+"IN QUESTO LUOGO, MA SAI CHE PER METTERTI IN "
1080 SN$=SN$+"SALVO DEVI RAGGIUNGERE IL VILLAGGIO CHE SI "
1090 SN$=SN$+"TROVA AL MARGINE DELLA FORESTA "
1100 GOSUB 5500: REM FORMATTAZIONE
1110 PRINT
1120 SN$="TI GUARDI ATTORNO CERCANDO LA GIUSTA DIREZIONE"
1130 GOSUB 5500: REM FORMATTAZIONE
1140 PRINT: PRINT "PREMI UN QUALSIASI TASTO PER GIOCARE"
1150 GET A$: IF A$="" THEN 1150
1160 PRINTCHR$(147): REM CANCELLA LO SCHERMO
1170 RETURN
1180 :
2000 REM ** DESCRIVE UN AMBIENTE **
2010 SN$="TI TROVI "+LN$(P): GOSUB 5500
2020 SN$="VEDI "
2030 REM ** VERIFICA INVENTARIO OGGETTI **
2040 F=0:SP$=""
2050 FOR I=1 TO 3
2060 IF VAL(IV$(I,2))<>P THEN 2080
2070 SN$=SN$+SP$+"A "+IV$(I,1):F=1:SP$=", "
2080 NEXT I
2090 IF F=0 THEN SN$=SN$+"NESSUN OGGETTO"
2100 GOSUB 5500: REM FORMATTA LA FRASE PER LO SCHERMO
2110 RETURN
2120 :
2299 :
2300 REM ** S/R INDICA LE USCITE **
2310 EX$=EX$(P)
2320 NR=VAL(LEFT$(EX$,2))
2330 ES=VAL(MID$(EX$,3,2))
2340 SD=VAL(MID$(EX$,5,2))
2350 OV=VAL(RIGHT$(EX$,2))
2353 :
2355 IF (NR OR ES OR SD OR OV)=0 THEN RETURN
2360 PRINT: SN$="LE USCITE POSSIBILI SONO A "
2370 IF NR <>0 THEN SN$=SN$+"NORD "
2380 IF ES <>0 THEN SN$=SN$+"EST "
2390 IF SD <>0 THEN SN$=SN$+"SUD "
2400 IF OV <>0 THEN SN$=SN$+"OVEST "
2410 GOSUB 5500: REM FORMATTA LA FRASE
2415 PRINT
2420 RETURN
2430 :
2500 REM ** S/R SEPARAZIONE COMANDI **
2510 IF IA$="ELENCA" OR IS$="FINE" THEN VB$=IS$:F=1: RETURN
2520 F=0
2530 LS=LEN(IS$)
2540 FOR C=1 TO LS
2550 A$=MID$(IS$,C,1)
2560 IF A$<>" " THEN 2590
2570 VB$=LEFT$(IS$,C-1):F=1
2580 NN$=RIGHT$(IS$,LS-C):C=LS
2590 RETURN
2600 :
2610 IF F=1 THEN RETURN
2620 PRINT: PRINT "MI SERVONO ALMENO DUE PAROLE"
2630 RETURN
2650 :
2700 REM ** S/R P LOCAZIONE SPECIALE? **
2705 SF=0: REM AZZERA FLAG PER LOCAZ. SPECIALI
2707 REM ** CREAZIONE CASUALE DI SPETTRI **
2710 IF P>4 AND RND (1)<0.1 THEN GOSUB 4290: RETURN
2715 :
2716 REM ** ALTRE LOCAZIONI SPECIALI **
2720 ON P GOSUB 4590, 4870, 5100, 4590
2730 RETURN
2735 :
3000 REM ** S/R COMANDI NORMALI **
3010 IF VF=0: REM FLAG DEI VERBI
3020 IF VB$="VAI" OR VB$="MOVI" THEN VF=1: GOSUB3500
3030 IF VB$="PRENDI" OR VB$"RACCOGLI" THEN VF=1: GOSUB 3700
3040 IF VB$="LASCIA" OR VB$"GETTA" THEN VF=1: GOSUB 3900
3050 IF VB$="ELENCA" OR VB$"INVENTARIO" THEN VF=1: GOSUB 4100
3055 IF VB$="GUARDA" THEN VF=1:MF=1: RETURN
3060 IF VB$="FINE" OR VB$="SMETTO" THEN VF=1: GOSUB 4170
3070 RETURN
3080 :
3500 REM ** S/R MOVIMENTO **
3505 GOSUB 3630: REM RICERCA DIREZIONE
3510 MF=1: REM SELEZIONA IL FLAG DI MOVIMENTO
3520 DR$=LEFT$(NN$,1)
3530 IF DR$ <>"N" AND DR$ <>"E" AND DR$ <>"S" AND DR$ <>"O" THEN
     GOTO 3590
3540 IF DR$="N" AND NR <>0 THEN P=NR: RETURN
3550 IF DR$="E" AND ES <>0 THEN P=ES: RETURN
3560 IF DR$="S" AND SD <>0 THEN P=SD: RETURN
3570 IF DR$="O" AND OV <>0 THEN P=OV: RETURN
3580 PRINT: PRINT "NON PUOI ANDARE A ";IS$
3585 MF=0: RETURN
3590 REM ** COMANDO DI DIREZIONE ERRATO **
3600 PRINT "CHE SIGNIFICA ";NN$;" ?"
3610 MF=0: RETURN
3620 :
3630 REM ** S/R RICERCA DIREZIONE **
3640 NN$=NN$+" ":LN=LEN(NN$):C=1
3645 FOR I=1 TO LN
3650 IF MID$(NN$,I,1)<>" " THEN NEXT I: RETURN
3655 W$=MID$(NN$,C,I-C):C=I+1
3660 IF W$="NORD" OR W$="EST" THEN NN$=W$:I=LN
3665 IF W$="SUD" OR W$="OVEST" THEN NN$=W$:I=LN
3670 NEXT I
3675 RETURN
3700 REM *** S/R PRENDI ***
3710 GOSUB 5300: REM OGGETTO VALIDO?
3720 IF F=0 THEN SN$ = W$ + " NON ESISTE": GOSUB 5500: RETURN
3730 OV=F: GOSUB 5450: REM VERIFICA INVENTARIO
3740 IF HF=1 THEN SN$=IV$(F,1) + " E' GIA' IN TUO POSSESSO":
     GOSUB 5500 : RETURN
3750 :
3755 REM ** L'OGGETTO E' QUI? ***
3760 IF VAL(IV$(F,2))<>P THEN SN$=IV$(F,1)+" NON E' QUI":
     GOSUB 5500: RETURN
3770 :
3780 REM *** AGGIUNGE OGGETTO ALL'ELENCO ***
3790 A=0
3800 FOR J=1 TO 2
3810 IF IC$(J)="" THEN IC$(J)=IV$(F,1):AF=1:J=2
3820 NEXT J
3830 :
3840 REM *** TROPPI OGGETTI ***
3850 IF AF=0 THEN PRINT "HAI GIA' DUE OGGETTI": RETURN
3860 :
3870 SN$="HAI RACCOLTO "+IV$(F,1): GOSUB 5500
3880 IV$(F,2)="-1": REM CANCELLA OGGETTO DA INVENTARIO
3890 RETURN
3895 :
3900 REM ** S/R LASCIA **
3910 GOSUB 5300: REM OGGETTO VALIDO?
3920 IF F=0 THEN SN$=W$ + " NON ESISTE": GOSUB 5500: RETURN
3930 :
3940 REM ** OGGETTO FRA QUELLI TRASPORTATI? **
3950 OV=F: GOSUB 5450
3960 IF HF=0 THEN SN$="NON HAI CON TE " + IV$(F,1): GOSUB 5500: RETURN
3970 :
3980 REM ** ABBANDONA L'OGGETTO **
3990 SN$="HAI ABBANDONATO "+IV$(F,1): GOSUB 5500
4000 IV$(F,2)=STR$(P): AGGIORNA INVENTARIO PRINCIPALE
4010 :
4020 REM ** RIMUOVE OGGETTO DA QUELLI TRASPORTATI **
4030 FOR J=1 TO 2
4040 IF IC$(J)=IV$(F,1) THEN IC$(J)="": J=2
4050 NEXT J
4060 RETURN
4070 :
4100 REM *** S/R VISUALIZZA INVENTARIO ***
4110 PRINT "OGGETTI TRASPORTATI:"
4120 FOR I=1 TO 2
4130 PRINT"  ";IC$(I)
4140 NEXT I
4150 RETURN
4160 :
4170 REM ** S/R DI FINE GIOCO **
4180 PRINT: PRINT"SEI SICURO (S/N)?"
4190 GET A$:IF A$<>"S" AND A$<>"N" THEN 4190
4200 IF A$="N" THEN RETURN
4210 END
4220 :
4290 REM ** S/R SPETTRO CASUALE **
4295 SF=1: GC=0
4300 SN$="BRIVIDI CORRONO LUNGO LA TUA SCHIENA"
4305 SN$=SN$+" IMPROVVISAMENTE UNA BIANCA APPARIZIONE"
4310 SN$=SN$+" SCIVOLA FUORI DALLA FORESTA E"
4315 SN$=SN$+" SI DIRIGE VERSO DI TE": GOSUB 5500
4320 :
4325 SN$="LO SPETTRO SI AVVICINA SEMPRE PIU'": GOSUB 5500
4330 GC=GC+1:IF GC>4 THEN GOSUB 4455: REM MORTE!
4335 PRINT: INPUT "IL TUO COMANDO";IS$
4340 GOSUB 2500: REM SUDDIVIDE IL COMANDO
4345 IF F=0 THEN 4325: REM COMANDO SUCCESSIVO
4350 OP=P: GOSUB 3000: REM ANALIZZA IL COMANDO
4355 IF MF=1 AND VB$="VAI" THEN GOSUB 4400: GOTO 4325
4357 IF MF=1 AND VB$="GUARDA" THEN GOSUB 2000: GOSUB 2300: GOTO 4325
4360 IF VF=1 THEN 4325: REM COMANDO SUCCESSIVO
4365 REM ** NUOVE PAROLE DI COMANDO **
4370 IF VB$="UCCIDI" OR VB$="LOTTA" THEN GOSUB 4425: GOTO 4325
4375 :
4385 IF VB$="CANTA" THEN GOSUB 4500: RETURN
4390 SN$="NON CAPISCO": GOSUB 5500: GOTO 4325
4395 :
4400 REM ** TENTATIVO DI MOVIMENTO **
4405 SN$="SEI PARALIZZATO DAL TERRORE E NON RIESCI"
4410 SN$=SN$+" A MUOVERTI ..!": MF=0: GOSUB 5500: P=OP
4415 RETURN
4420 :
4425 REM ** LOTTA O UCCISIONE **
4430 SN$="LO SPETTRO E' UN ESSERE SOPRANNATURALE"
4435 SN$=SN$+" E SI BEFFA DEI TUOI DEBOLI TENTATIVI"
4440 SN$=SN$+" DI COLPIRLO": GOSUB 5500
4445 RETURN
4450 :
4455 REM ** MORTE **
4460 SN$="IL DOLORE AL PETTO DIVENTA INSOPPORTABILE"
4465 SN$=SN$+" E CADI PESANTEMENTE AL SUOLO.": GOSUB 5500
4470 SN$="IL TUO SPIRITO SI STACCA DAL CORPO INERTE"
4475 SN$=SN$+" E FLUTTUA VIA, SCOMPARENDO NELLA NEBBIA."
```

KEVIN JONES

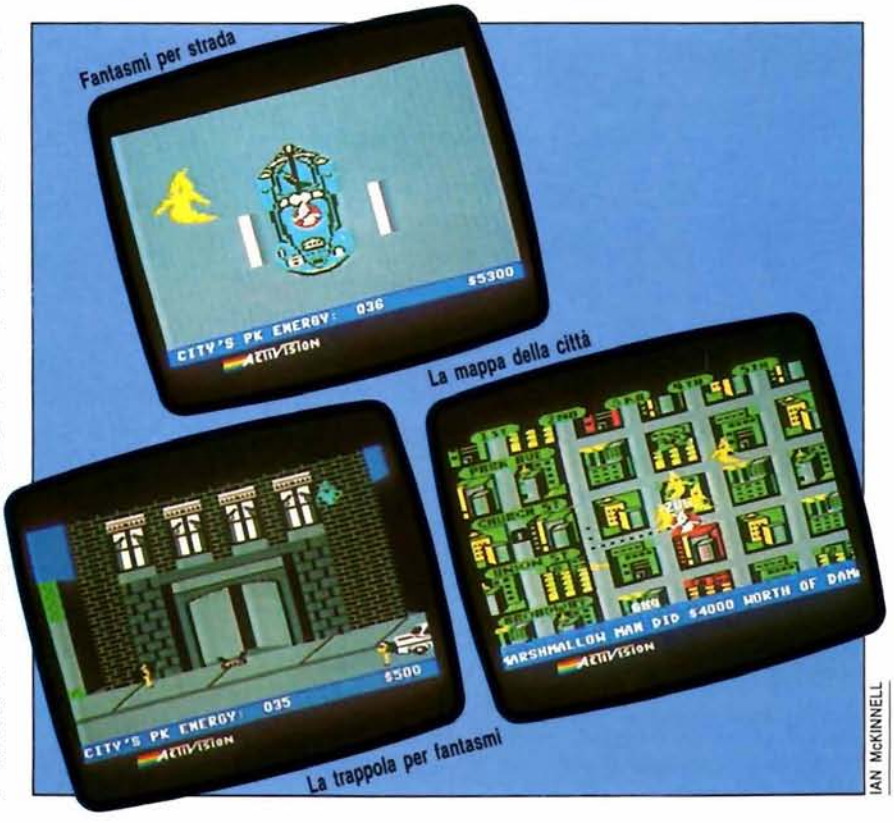
```
4480 SN$=SN$+" PER UNIRSI ALLE ALTRE ANIME TORMENTATE"
4485 SN$=SN$+" DELLA FORESTA INCANTATA.": GOSUB 5500
4490 END
4495 :
4500 ** CANTO DI SALVEZZA **
4505 SN$="CONOSCI SOLO TRE TEMI DI FILM:": GOSUB 5500
4510 SN$="1) GHOSTBUSTERS": GOSUB 5500
4515 SN$="2) IL FANTASMA DELL'OPERA": GOSUB 5500
4520 SN$="3) FANTASMI A ROMA": GOSUB 5500
4525 PRINT:INPUT "QUALE SCEGLI (1/2/3)"; C$
4530 IF VAL(C$)>3 OR VAL(C$)<1 THEN PRINT: PRINT "RIPETI!": GOTO 4525
4535 CR=INT(RND(1)*3)+1
4537 IF CR<>VAL(C$) THEN GOSUB 4542: REM CANZONE SBAGLIATA!
4540 GOSUB 4565: RETURN: REM CANZONE GIUSTA!
4542 REM ** S/R CANZONE SBAGLIATA **
4545 SN$="LO SPETTRO NUTRE UNA PARTICOLARE AVVERSIONE"
4550 SN$=SN$+" PER QUESTA CANZONE E TI COLPISCE.": GOSUB 5500
4555 GOSUB 4455: REM MORTE
4560 :
4565 REM ** CANZONE GIUSTA **
4570 SN$="LO SPETTRO E' PLACATO DAL TUO CANTO E"
4575 SN$=SN$+" SI DISSOLVE NELLA BRUMA DELLA FORESTA": GOSUB 5500
4580 RETURN
4585 :
4590 ** S/R INGRESSO ALLA GALLERIA **
4600 SF=1
4605 SN$="SEI GIUNTO ALL'INGRESSO DI UNA LUNGA GALLERIA": GOSUB 5500
4610 SN$="PUOI ENTRARE O TORNARE INDIETRO SUL  SENTIERO": GOSUB 5500
4620 :
4625 PRINT: INPUT "IL TUO COMANDO";IS$
4630 GOSUB 2500: REM DIVIDE I COMANDI
4635 IF F=0 THEN 4625: REM COMANDO NON VALIDO
4637 GOSUB 3000: REM COMANDO NORMALE
4640 IF MF=1 THEN RETURN: REM GIOCATORE INDIETREGGA
4645 IF VF=1 THEN 4625: REM COMANDO ESEGUITO
4650 REM ** NUOVI COMANDI **
4655 IF VB$="ENTRA" THEN GOSUB 4700: RETURN
4660 IF VB$="TORNA" AND P=4 THEN MF=1: P=6: RETURN
4665 IF VB$="TORNA" AND P=1 THEN MF=1: P=9: RETURN
4667 SN$="NON CAPISCO": GOSUB 5500: GOTO 4625
4700 REM ** INGRESSO NELLA GALLERIA **
4705 SN$="ENTRI NELLA GALLERIA MA TI TROVI NEL BUIO PIU' COMPLETO"
4710 SN$=SN$+" E NON VEDI USCITE.": GOSUB 5500
4725 PRINT:INPUT "IL TUO COMANDO";IS$
4730 GOSUB 2500: REM SUDDIVIDE IL COMANDO
4732 :
4735 IF F=0 THEN 4725: COMANDO NON VALIDO
4740 OP=P:GOSUB 3000: REM COMANDI NORMALI
4745 IF MF=1 THEN SN$="E' TANTO BUIO CHE RIESCI A SCORGERE SOLO ":P=OP
4747 IF MF=1 THEN SN$=SN$+" L'INGRESSO DELLA GALLERIA": GOSUB 5500:
     MF=0: GOTO 4725
4750 IF VF=1 THEN 4725: REM COMANDO ESEGUITO
4755 IF VB$="TORNA" AND P=4 THEN MF=1: P=6: RETURN
4760 IF VB$="TORNA" AND P=1 THEN MF=1: P=9: RETURN
4762 IF VB$<>"USA" AND VB$<>"ACCENDI" THEN SN$="NON CAPISCO"
4765 IF VB$<>"USA" AND VB$<>"ACCENDI" THEN GOSUB 5500: GOTO 4725
4777 :
4780 REM ** CERCA LA LANTERNA **
4790 GOSUB 5300: REM OGGETTO VALIDO?
4795 OV=F: GOSUB 5450: REM IN POSSESSO DEL GIOCATORE?
4797 IF F=0 THEN SN$=W$+" NON ESISTE": GOSUB 5500: GOTO 4725
4800 IF HF=0 THEN SN$=" NON POSSIEDI " + IV$(F,1): GOSUB 5500: GOTO 4725
4810 REM ** L'OGGETTO E' LA LANTERNA? **
4815 IF F<>2 THEN SN$="QUI "+IV$(F,1)+" E' INUTILE": GOSUB 5500:
     GOTO 4725
4835 REM ** SUCCESSO! **
4840 SN$="USI LA LANTERNA PER ILLUMINARE LA GALLERIA"
4845 SN$=SN$+" E FINALMENTE RAGGIUNGI L'USCITA.": GOSUB 5500
4850 IF P=1 THEN MF=1: P=4: RETURN
4855 IF P=4 THEN MF=1: P=1: RETURN
4860 :
4870 REM ** S/R PALUDE **
4875 SF=1
4880 SN$="COMINCI AD AFFONDARE IN UNA PALUDE.": GOSUB 5500
4885 PRINT: INPUT "COMANDI";IS$
4890 GOSUB 2500: REM SUDDIVIDE IL COMANDO
4895 IF F=0 THEN 4885: REM COMANDO NON VALIDO
4900 GOSUB 3000: REM COMANDO NORMALE
4910 IF VB$="GUARDA" THEN GOSUB 2000: GOTO 4885
4915 IF VB$="LASCIA" THEN IV$(F,2)="-2": REM OGGETTO PERDUTO PER SEMPRE
4917 IF VF=1 THEN 4885: REM COMANDO NORMALE
4920 REM ** NUOVI COMANDI **
4925 IF VB$<>"NUOTA" THEN SN$="NON CAPISCO": GOSUB 5500: GOTO 4885
4930 REM ** NUOTATA **
4932 F=0
4935 FOR I=1 TO 2
4940 IF IC$(I) <>"" THEN F=F+1
4950 NEXT I
4955 IF F<2 THEN GOSUB 5035: RETURN: REM CONTINUA A NUOTARE
4960 GOSUB 5000: RETURN: REM POSSIEDE DUE OGGETTI
5000 REM ** S/R POSSIEDE DUE OGGETTI **
5010 SN$="GLI OGGETTI TI APPESANTISCONO E TI TRASCINANO A FONDO.":
     GOSUB 5500
5012 PRINT: INPUT "COMANDI";IS$
5015 GOSUB 2500: REM SUDDIVIDE IL COMANDO
5020 IF VB$<>"LASCIA" THEN GOSUB 5080: REM AFFONDA
5025 GOSUB 3900: IV$(F,2)="-2": REM LASCIA UN OGGETTO
5030 IF HF=0 OR F=0 THEN 5080: REM AFFONDA
5035 REM ** CONTINUA A NUOTARE **
5040 SN$="ADESSO PUOI ATTRAVERSARE A NUOTO LA PALUDE. QUALE DIREZIONE
     SCEGLI ?": GOSUB 5500
5050 EX$(2)="00080605": GOSUB 2300: REM DEFINISCE E VISUALIZZA LE USCITE
5055 PRINT: INPUT "COMANDI";IS$
5060 GOSUB 2500: REM SUDDIVIDE IL COMANDO
5062 IF F=0 THEN 5055: REM COMANDO NON VALIDO
5065 GOSUB 3500: REM MUOVE
5067 EX$(2)="00000000": REM NESSUNA USCITA
5070 RETURN
5075 :
5080 REM ** S/R AFFONDA **
5085 SN$="AFFONDI NELLA PALUDE E ANNEGHI": GOSUB 5500
5090 END
5100 REM ** S/R VILLAGGIO **
5102 SF=1
5105 SN$="IL VILLAGGIO E' PROTETTO DA UN ALTO MURO.": GOSUB 5500
5106 IF GF<>0 THEN GOSUB 5190: RETURN: REM PORTA
5107 SN$="UNA GUARDIA SORVEGLIA L'UNICA PORTA D'ACCESSO": GOSUB 5500
5115 PRINT: INPUT "COMANDI";IS$
5120 GOSUB 2500: IF F=0 THEN 5115: REM COMANDO NON VALIDO
5125 GOSUB 3000: REM COMANDI NORMALI
5130 IF VB$="GUARDA" THEN GOSUB 2000: REM DESCRIVE
5135 IF VB$="VAI" AND MF=1 THEN RETURN
5140 IF VF=1 THEN 5115: REM COMANDO SUCCESSIVO
5145 IF VB$<>"UCCIDI" THEN SN$="NON CAPISCO": GOSUB 5500: GOTO 5115
5150 REM ** UCCIDE **
5155 SN$="COSA USI PER UCCIDERE LA GUARDIA ?": GOSUB 5500
5160 SN$="SCRIVERE IL NOME DELL'OGGETTO OPPURE <I> PER  ISTRUZIONI":
     GOSUB 5500
5162 INPUT IS$:IF IS$="I" THEN 5115
5165 GOSUB 2500: REM SUDDIVIDE
5167 IF F=0 THEN 5160: REM COMANDO NON VALIDO
5170 GOSUB 5300: IF F=0 THEN SN$="NON HAI "+W$: GOSUB 5500: GOTO 5160
5172 OV=F: GOSUB 5450: REM OGGETTO POSSEDUTO?
5174 IF HF=0 THEN SN$="NON POSSIEDI "+IV$(F,1): GOSUB 5500: GOTO 5160
5175 IF F<>1 THEN SN$=IV$(F,1)+" NON TI E' DI ALCUN AIUTO":
     GOSUB 5500: GOTO 5160
5180 SN$="LA GUARDIA E' STATA UCCISA": GOSUB 5500: GF=1
5185 :
5190 REM ** S/R PORTA CHIUSA **
5195 SN$="TI AVVICINI E CERCHI DI APRIRE LA PORTA DEL VILLAGGIO"
5200 SN$=SN$+" MA LA PORTA E' CHIUSA E NON SI APRE": GOSUB 5500
5205 PRINT: INPUT "COMANDI";IS$
5210 GOSUB 2500: IF F=0 THEN 5205: REM COMANDO NON VALIDO
5215 GOSUB 3000: REM COMANDO NORMALE
5220 IF VB$="GUARDA" THEN GOSUB 2000: REM DESCRIVE
5225 IF VB$="VAI" AND MF=1 THEN RETURN
5230 IF VF=1 THEN 5205: REM COMANDO SUCCESSIVO
5232 IF VB$="USA" THEN 5240
5234 IF VB$="APRI" THEN SN$="COME ?": GOSUB 5500: GOTO 5205
5235 SN$="NON CAPISCO": GOSUB 5500: GOTO 5205
5240 GOSUB 5300: REM OGGETTO VALIDO
5242 OV=F: GOSUB 5450: REM OGGETTO POSSEDUTO?
5244 IF F=0 THEN SN$=W$+" NON ESISTE!": GOSUB 5500 :GOTO 5205
5246 IF HF=0 THEN SN$="NON POSSIEDI IL "+IV$(F,1): GOSUB 5500: GOTO 5205
5248 IF F<>3 THEN SN$=IV$(F,1)+" NON TI E' DI ALCUN AIUTO": GOSUB 5500:
     GOTO 5205
5250 REM ** AL SICURO! **
5255 SN$="HAI APERTO LA PORTA E, INDOSSATI GLI ABITI DELLA"
5260 SN$=SN$+" GUARDIA UCCISA, ATTRAVERSI IL VILLAGGIO SENZA ESSERE
     RICONOSCIUTO."
5265 SN$=SN$+"FINALMENTE SEI IN SALVO !": GOSUB 5500
5270 END
5299 :
5300 REM *** S/R OGGETTO VALIDO? ***
5310 NN$=NN$+" ":LN=LEN(NN$):C=1:F=0
5315 FOR K=1 TO LN
5320 IF MID$(NN$,K,1)<>" " THEN NEXT K: RETURN
5325 W$=MID$(NN$,C,K-C):C=K+1
5330 LW=LEN(W$)
5335 FOR J=1 TO 3
5340 LI=LEN(IV$(J,1)): REM LUNGHEZZA OGGETTO
5350 FOR I=1 TO LI-LW+1
5360 IF MID$(IV$(J,1),I,LW)=W$ THEN F=J:I=LI:J=3:K=LN
5370 NEXT I,J,K
5380 RETURN
5390 :
5450 REM *** S/R OGGETTO TRASPORTATO? ***
5460 HF=0
5470 IF IV$(OV,2)="-1"THEN HF=1
5480 RETURN
5490 :
5500 ** S/R VISUALIZZ. FORMATTA **
5510 LC=01:          REM CONTATORE CAR/LINEA
5520 OC=1:           REM VAL. INIZIALE CONTO VECCHIO
5530 OW$="":         REM VAL. INIZIALE PAROLA VECCHIA
5540 LL=40:          REM LUNGHEZZA LINEA
5550 SN$=SN$+"       DUMMY "
5560 PRINT
5570 FOR C=1 TO LEN(SN$)
5580 LC=LC+1
5590 IF ID$(SN$,C,I)=" " THEN GOSUB 5000
5600 NEXT C
5610 RETURN
5620 :
5800 REM ** S/R  VERIFICA FINE LINEA **
5810 NW$=MID$(SN$,OC,C-OC+1):REM NEW WORD
5820 IF LC<LL THEN PRIN OW$; : GOTO 5840
5830 PRINT OW$:LC=LEN(NW$)
5840 OC=C+1:OW$=NW$
5850 RETURN
6000 REM ** LETTURA DATI DEGLI OGGETTI E DELLA MAPPA **
6010 DIM IV$(3,2),LN$(10),EX$(10),IC$(2)
6020 FOR C=1 TO 3
6030 READ IV$(C,1),IV$(C,2)
6040 NEXT C
6050 :
6060 FOR C=1 TO 10
6065 READ LN$(C),EX$(C)
6070 CC=CC+VAL(EX$(C)): REM TOTALE CHECKSUM
6080 NEXT C
6090 :
6100 READ CD:IFCD<>CC THEN PRINT"ERRORE CHECKSUM":STOP
6110 :
6120 REM ** DATI PER GLI OGGETTI **
6130 DATA PISTOLA,10,TORCIA,9,CHIAVE,5
6140 :
6150 REM ** DATI PER LA MAPPA **
6160 DATA VICINO ALL'ENTRATA DI UN TUNNEL,00000900
6170 DATA IN UNA PALUDE,00000000
6180 DATA IN UN VILLAGGIO,07000000
6190 DATA VICINO ALL'INGRESSO DI UN TUNNEL,05060000
6200 DATA SU UN SENTIERO,00020400
6210 DATA SU UN SENTIERO,02070004
6220 DATA SU UN SENTIERO,08000306
6230 DATA SU UN SENTIERO,09000702
6240 DATA SU UN SENTIERO,01100800
6250 DATA IN UNA RADURA,00000009
6260 REM ** DATO PER CHECKSUM **
6270 DATA 32253121
6280 RETURN
```

Ghostbusters

La trama di alcuni giochi per computer deriva da quella di film di successo

Il gioco *Ghostbusters*, prodotto dalla Activision, prende largo spunto dall'omonimo film, che ha riscosso un grandissimo successo di pubblico.
Il pacchetto, sviluppato in collaborazione con la Columbia Pictures, la casa produttrice del film, segue molto da vicino la sua sceneggiatura e ne riprende la colonna sonora che, nell'estate del 1984, è stata in testa alle classifiche di vendita.
Ghostbusters narra la storia di tre ricercatori che fondano a New York un'agenzia per la caccia ai fantasmi. Essi scoprono che schiere di spettri si stanno radunando in un palazzo della metropoli, con l'intento di causare la fine del mondo. All'inizio del gioco, il giocatore riceve una somma di denaro con la quale acquistare un'automobile e l'armamentario per l'annientamento dei fantasmi. Le sofisticate apparecchiature "anti-spettro" comprendono un "rivelatore d'energia PK" per l'individuazione dei fantasmi, un "aspira-fantasmi" per la loro cattura e una "rete" a raggi laser con la quale i fantasmi vengono spinti in una trappola mortale. Acquistati gli apparecchi, il giocatore deve avventurarsi per le strade di New York guadagnandosi da vivere con la cattura dei fantasmi.
Lo scenario in cui si svolge questa parte del gioco è la mappa della città. Al suo centro compare il palazzo nel quale si stanno radunando i fantasmi, attirati dal malvagio Zuul. Gli spettri compaiono in varie zone della metropoli e il giocatore deve dirigere l'auto, chiamata appunto "Ghostbuster", verso le aree infestate. Durante l'attraversamento della città, a fianco della vettura appaiono dei fantasmi chiamati "Roamers". Se il giocatore ha caricato sul tetto dell'auto l'"aspira-fantasmi", può catturare i Roamers e guadagnare punti.
Raggiunte le zone infestate, il giocatore può montare le proprie apparecchiature: per prima cosa piazza una trappola al centro dello schermo, quindi cerca di spingere i fantasmi verso la trappola usando la rete a raggi laser. Quando il fantasma si trova sopra la trappola, quest'ultima scatta imprigionandolo. Allora la colonna sonora del gioco si interrompe per un istante e una voce si congratula col giocatore pronunciando la fatidica parola «Ghostbuster!».
Durante la fase di cattura, il giocatore deve sempre tenere d'occhio il segnale d'allarme «Marshmallow», che indica se i fantasmi sono riusciti a radunarsi, dando così vita (si fa per dire!) al terribile Marshmallow Man. Questo strano essere minaccia di distruggere la città se non vengono immediatamente ritirate alcune trappole. Inoltre il giocatore deve evitare che altre due entità, Gatekeeper e Keymaster, congiungano le proprie forze all'interno del Tempio di Zuul. Se la fusione avviene prima di aver raggiunto un punteggio sufficiente ad aprire le porte del Tempio ai tre cacciatori di fantasmi, la partita è persa.
Ghostbusters è un gioco progettato con molta cura, dotato di un'ottima grafica e di una piacevole colonna sonora. L'esclamazione pronunciata ad ogni cattura aggiunge un tocco di realismo allo sviluppo dell'azione e la caccia ai fantasmi tiene "incollati" allo schermo per ore anche i più smaliziati appassionati di giochi per computer.

Ghostbusters: Per il Commodore 64 e lo Spectrum Sinclair
Editori: Activision Inc, (USA)
Autori: David Crane, Adam Bellin, Hilary Mills.
Joystick: Necessario
Formato: Su cassetta



Fantasmi ... nel computer
Le tre immagini sono tratte dal gioco Ghostbuster della Activision. Il giocatore, percorrendo le strade della città, viene affiancato dai fantasmi che possono essere catturati grazie all'apparecchio "aspira-fantasmi" montato sul tetto dell'automobile.
In un'altra fase del gioco, i fantasmi vengono spinti verso una trappola mortale usando una rete a raggi laser. La mappa mostra l'ubicazione del Tempio di Zuul e segnala la presenza di Marshmallow Man, che minaccia di distruggere l'intera città

Il computer insegna

Introdurre l'uso di computer nelle scuole è un processo che richiede tempo



TONY LODGE

Computer contro lavagna
In una moderna aula scolastica, un computer dotato di software adeguato può svolgere le funzioni di lavagna, proiettore, televisore, registratore a cassette e, anche, di biblioteca. Alternativamente, il computer può essere impiegato come supporto didattico secondario nell'insegnamento tradizionale, sollevando gli insegnanti da una parte del gravoso lavoro di correzione dei compiti, ad esempio

«Nel prossimo futuro l'introduzione dei computer nei sistemi d'informazione rivoluzionerà le tecniche didattiche al pari dell'invenzione della stampa». Questa affermazione, tratta da un rapporto governativo pubblicato in Gran Bretagna nel 1978 e successivamente ripresa dal professor Tom Stonier, un'autorità nel campo delle nuove tecnologie, è stata ribadita anche da Seymour Papert, noto esperto statunitense di pedagogia. Egli ha affermato che la scuola, come oggi la conosciamo, è destinata a scomparire in di poche decine d'anni. In realtà, qualsiasi trasformazione avviene con una certa lentezza, che dipende, nonostante una certa dose di scetticismo soprattutto nel corpo insegnante, più da ragioni amministrative che non dalla disponibilità di nuovi sussidi didattici ed esperimenti didattici risalgono addirittura alla fine degli anni '20 e alcuni dei principi alla base di tali esperienze mantengono ancor oggi un'indiscussa validità.

Già nel 1920 vennero realizzate semplici macchine didattiche, che tuttavia divennero popolari solo intorno al 1950, quando B. F. Skinner, uno dei maggiori studiosi di psicologia dell'apprendimento degli anni '50 e '60, fabbricò un primitivo apparecchio didattico interattivo: studiando il comportamento di topi e di piccioni, mirava a realizzare un congegno di puntamento per missili "pilotato", se così si può dire, da tre piccioni viaggiatori collocati nella sua punta: un progetto che anticipava, concettualmente, di circa 20 anni il missile americano Cruise. Adoperando il cibo come meccanismo di condizionamento

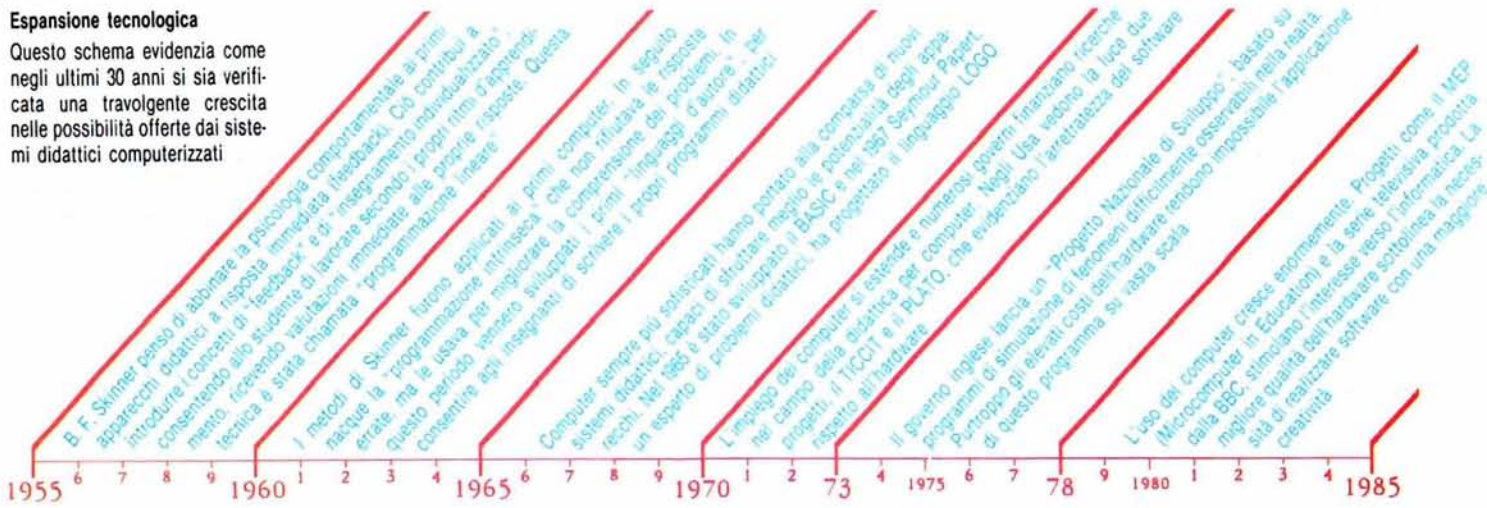
pa. Nonostante il Pentagono, respingesse il progetto di Skinner, i suoi studi sull'apprendimento riscossero un notevole successo. Egli aveva sviluppato un sistema di "programmazione lineare" (che non ha alcuna relazione con la programmazione di un computer) in base al quale le informazioni venivano comunicate allo studente in "piccole dosi" e ampliate solo dopo ogni risposta esatta. In definitiva Skinner intendeva forse applicare agli studenti le stesse tecniche adottate per i "piccioni pilota"!

La programmazione lineare

Da questa tecnica ebbero origine due concetti fondamentali: il "feedback" (o informazione di ritorno) e la didattica "mirata". Il feedback si riferisce all'immediata comunicazione data allo studente sulla correttezza della risposta fornita a un particolare quesito. Purtroppo, però, gli apparecchi di Skinner accettavano solo una limitata gamma di risposte. La didattica mirata riguarda invece la natura "diretta" del metodo d'insegnamento, che permette allo studente di lavorare secondo i propri ritmi di apprendimento e non secondo quelli stabiliti dall'insegnante.
Nel momento di maggior popolarità delle macchine didattiche di Skinner, iniziarono a diffondersi anche i computer, quindi i sistemi di programmazione lineare vennero ben presto applicati anche su questo "nuovo" tipo di apparecchiatura. Nonostante i notevoli limiti, queste tecniche vengono ancora oggi impiegate da numerosi programmi: messaggi del tipo Spiacente, riprova ancora... e Giusto, risposta esatta! compaiono sugli schermi dei computer per interrompere la tediosità di una sequenza di domande.
Dalla programmazione lineare sono derivati i cosiddetti "programmi diramati", che non si limitano soltanto ad offrire allo studente la possibilità di una seconda risposta, nel caso che la prima non sia corretta, ma lo aiutano a capire l'errore fornendo adeguate indicazioni; questa nuova tecnica tiene maggior conto delle caratteristiche individuali dello studente.
In un secondo momento sono apparsi i "linguaggi d'autore", progettati per consentire agli insegnanti di redigere i propri programmi. Essi rappresentano un primo tentativo di rendere l'approccio alla programmazione più "amichevole" e di permettere agli insegnanti di sviluppare programmi didattici senza obbligarli ad apprendere la vera e propria programmazione di un computer.
Tuttavia, sia la programmazione lineare sia quella intrinseca hanno grossi limiti. Innanzi tutto considerano lo studente un "vaso vuoto" da riempire di nozioni e l'apprendimento una semplice acquisizione di fatti e non un frutto dell'esperienza. Non concedendo alcuno spazio alla fantasia e alla creatività, spingono lo studente a dare risposte corrette ma schematiche. La cattiva fama creatasi attorno a questi sistemi, noti genericamente come "apprendimento programmato", ha influenzato in seguito anche qualsiasi tipo di insegnamento basato su computer.
Considerate queste premesse, i computer hanno inizialmente risentito dei limiti dei primi apparecchi didattici, ma ben presto si sono dimostrati capaci di aprire nuove aree di sviluppo nel campo dell'insegnamento. Molte delle esperienze sviluppate inizialmente in questo campo sono ancora oggi valide e non solo per gli insegnanti nell'ambito scolastico. Un progetto sviluppato nel 1956 al Dartmouth College, negli Stati Uniti, mirava a creare un linguaggio d'insegnamento il più vicino possibile alla lingua parlata (inglese, ovviamente), che consentisse a chiunque di scrivere programmi per computer. Considerando fattori limitativi dovuti all'hardware del tempo (un apparecchio con 16 Kbyte di memoria costava 40.000 dollari e occupava un'intera stanza), il linguaggio doveva prevedere il minimo ingombro di memoria possibile. Il risultato fu il "Beginner's All-purpose Symbolic Instruction Code", ossia il BASIC.
Due anni dopo la comparsa del BASIC, un gruppo di studiosi d'informatica e di tecniche didattiche del Massachusetts Institute of Technology (MIT) sviluppò un linguaggio di programmazione di grande semplicità che permetteva anche ai bambini di programmare un computer. L'obiettivo era veramente ambizioso per quel tempo, perché i linguaggi precedenti erano tutti destinati a studenti di

Espansione tecnologica

Questo schema evidenzia come negli ultimi 30 anni si sia verificata una travolgente crescita nelle possibilità offerte dai sistemi didattici computerizzati



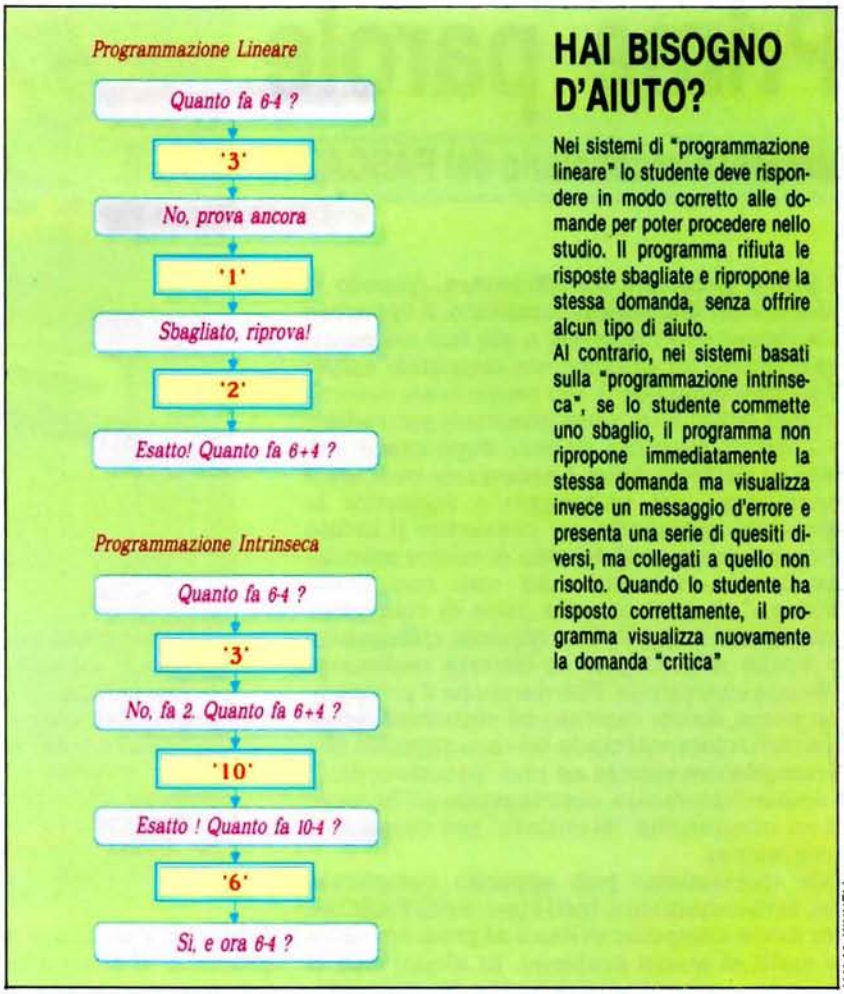
La comparsa dei computer nelle aule scolastiche non è un fatto nuovo: alcuni progetti

Skinner aveva addestrato alcuni piccioni a dirigersi verso un preciso punto di una map-

livello universitario. Il direttore del progetto era Seymour Papert, che aveva studiato con il famoso psicologo dell'età infantile Piaget, che riteneva necessario creare un ambiente che esaltasse il desiderio dei bambini di scoprire e imparare. Il frutto di questo lavoro d'equipe fu il LOGO, un linguaggio che oggi correda molti micro e che viene comunemente insegnato nelle scuole elementari d'oltre oceano. L'avvento del BASIC e del LOGO ha quindi avvicinato alla programmazione dei computer anche gli studenti più giovani.
Nonostante le energie spese nel perfezionamento di questi sistemi didattici pilotati da computer, molte persone continuano a nutrire forti dubbi sull'impiego del computer nella scuola. Per fugare queste incertezze, la National Science Fundation of America ha investito 10 milioni di dollari in due progetti volti ad approfondire gli aspetti del problema: il TICCIT e il PLATO.

Il TICCIT e il PLATO

Obiettivo del TICCIT (Time-shared Interactive Computer Controlled Information Television) era quello di fornire un insegnamento di qualità migliore e a costi meno elevati rispetto a quello tradizionale dei "college". La fondazione commissionò a una fabbrica di computer la realizzazione di hardware e software didattici e la progettazione di una tastiera dotata di speciali pulsanti per il controllo dell'apprendimento, di uno schermo televisivo (adatto per i testi, la grafica e proiezioni di videonastri) e di un altoparlante per i messaggi audio. Il sistema realizzato era composto da 128 terminali, controllati da due microcomputer. Purtroppo non tutti i terminali funzionarono a dovere, quindi molti studenti non conclusero il corso, ma gli altri ottennero risultati sensibilmente superiori rispetto agli studenti che seguivano corsi di tipo tradizionale. Il sistema TICCIT è stato successivamente introdotto in due "college pilota", ma non su scala nazionale.
Il PLATO (Programmed Logic for Automatic Teaching Operation) si basava invece su una rete di circa 1.000 terminali, collegati ad una unità d'elaborazione centrale situata in Illinois. I terminali potevano comunicare tra loro e accedere ad una vasta raccolta di lezioni, contenuta nella memoria centrale. I terminali video del PLATO possedevano speciali pannelli trasparenti che permettevano di proiettare diapositive colorate sulla grafica. Un pannello sensibile al tatto consentiva agli studenti di comunicare col computer semplicemente toccando lo schermo e, inoltre, gli insegnanti potevano scrivere i propri programmi in un "linguaggio d'autore" chiamato TUTOR. Come sul TICCIT, la preparazione del software richiedeva meno tempo di quanto si potrebbe pensare. I risultati ottenuti dagli studenti che usavano il PLATO furono pari a quelli dei ragazzi impegnati nei "college" convenzionali, ma gli studenti del PLATO giudicarono questo sistema più "amichevole" rispetto a quelli che avevano studiato sul TICCIT.



Nel 1973, il governo inglese finanziò un Programma Nazionale di Sviluppo dedicato all'apprendimento pilotato da computer. Gran parte del finanziamento è stata spesa per sviluppare programmi di simulazione di vario genere, mentre il computer è stato usato per produrre eventi la cui osservazione è difficile o impossibile nella realtà. Un programma, ad esempio, simulava la concentrazione di colorante in uno striscio di sangue a diversi intervalli di tempo e reazioni chimiche di altro tipo. Tuttavia gran parte del software mancava di originalità e gli alti costi dei computer, nel 1970, rendevano difficilmente realizzabile l'impiego del sistema su larga scala.
Qualsiasi progetto didattico, purtroppo, è condizionato da problemi politici, economici e sociali. L'impiego dei computer come strumenti didattici, oltre a tener conto di questi fattori e dei continui sviluppi della tecnologia, deve superare la resistenza di chi considera i computer una semplice estensione delle discusse tecniche di "apprendimento programmato" descritte in precedenza.
D'altra parte, come tutti gli altri campi dell'informatica, anche la didattica computerizzata trae profitto dal rapido calo del costo delle memorie e quindi dell'hardware in generale.

Prime parole

Sintassi e vocabolario del PASCAL

Il primo problema da affrontare, quando si adopera un linguaggio compilato, è imparare a lavorare con i processi a più fasi necessari ad ottenere un programma eseguibile anche di piccole dimensioni. Per prima cosa, occorre un text editor o un word processor per redigere il listato sorgente. Quindi, dopo averlo salvato su nastro o disco, è necessario caricare il compilatore del linguaggio e impartire le istruzioni necessarie per convertire il listato sorgente in un qualche tipo di codice macchina (spesso ricorrendo ad una complicata "stringa" di opzioni nella linea di comando). Infine il file "oggetto" va rilocato, collegandolo anche alle routine di libreria necessarie alla sua esecuzione. Può darsi che il programma possa essere caricato ed elaborato senza ulteriori interventi ma in tal caso significa che il compilatore ricorre ad uno "pseudo-codice" o codice intermedio, costringendo all'impiego di un programma "esecutore" per eseguire il programma.

Tale meccanismo può apparire complicato ma, fortunatamente, tutti i pacchetti PASCAL per home computer evitano al programmatore molti di questi problemi. In alcuni casi si trovano contemporaneamente in memoria, addirittura, il codice sorgente, il compilatore e lo stesso programma oggetto. Ciò è possibile grazie alla discreta efficienza e alle ridotte dimensioni del PASCAL: il funzionamento del sistema risultante è spesso più semplice di quello di un package in BASIC residente. Solo nel caso di programmi di notevoli dimensioni diviene necessario ricorrere ai complicati processi descritti in precedenza.

Ogni compilatore prevede una propria serie di comandi, spiegati nel relativo manuale, per far funzionare il programma di editor e il compilatore. Spesso bastano le tre lettere di comando E, C e R rispettivamente per redigere (Edit), compilare (Compile) e lanciare (Run) un programma. Ma ciò che conta di più è l'uso di una corretta sintassi del linguaggio, indipendentemente dalla complessità del programma sviluppato. Il PASCAL ha raggiunto un grado di standardizzazione tale, che solo in rari casi si rendono necessarie modifiche ai programmi, contrariamente a quanto avviene per i diversi dialetti del BASIC.

Passiamo adesso ad esaminare il nostro primo programma completo in PASCAL:

```
Program ProgrammaUno (output);
Const
    Messaggio = 'Programmazione in Pascal';
Begin
    write (Messaggio)
End.
```



Per qualunque tasca
Un package professionale del linguaggio PASCAL può costare anche più di un home computer, ma negli ultimi tempi sono stati messi in commercio, a prezzo ragionevole, alcuni compilatori adatti ai più diffusi home computer. La figura mostra alcuni dei pacchetti disponibili

Prima di analizzare in dettaglio questo programma, si provi a inserirlo in memoria e a compilarlo, prestando attenzione agli eventuali messaggi d'errore comunicati dal compilatore e tentando di individuare gli errori commessi. Ogni simbolo deve essere trascritto così come appare nel listato, con particolare attenzione ai punti e virgola posti alla fine della prima e della terza linea e al punto della linea finale. Se il programma viene compilato in modo corretto, alla sua esecuzione sullo schermo appare il messaggio:

```
Programmazione in Pascal
```

Il programma di per sé è veramente banale, ma evidenzia la struttura generale tipica di qualsiasi modulo PASCAL (sia esso un programma, una procedura o una funzione) che viene suddiviso in tre parti:

1. Il titolo, in questo caso il nome di un programma.
2. Le dichiarazioni e le definizioni. In questo caso c'è la sola definizione di una costante.
3. Il "corpo" del modulo, che contiene tutte le istruzioni eseguibili.

Per meglio chiarire le caratteristiche sintattiche del PASCAL, almeno per quanto riguarda le sue basi, si può ricorrere a "schemi sintattici", ossia diagrammi che si sviluppano a senso unico verso il basso e da sinistra verso destra. I blocchi che li compongono rappresentano "entità sintattiche" contenute in riquadri ad angoli arrotondati oppure un altro elemento, descritto a sua volta in diagrammi

Pacchetto: HISOFT PASCAL
Per: Spectrum, Amstrad, apparecchi MSX
Caratteristiche: È una versione non esattamente standard, ma di buona qualità. È corredata di un proprio editor e di una libreria di "grafica della tartaruga"

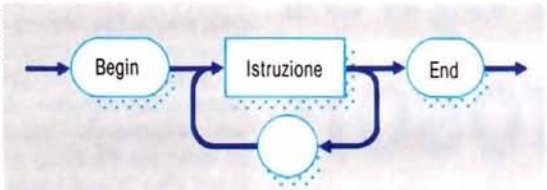
Pacchetto: ISO PASCAL della Acorn
Per: BBC Modello B, Electron
Caratteristiche: Questo è un ottimo affare; possiede due compilatori: uno, residente in ROM e dotato di un editor semi-intelligente, consente la compilazione "da memoria a memoria"; l'altro, su dischetto, è dedicato a chi possiede la scheda con il secondo processore

Pacchetto: Oxford PASCAL
Per: Commodore 64
Caratteristiche: Anche questo è un eccellente package. Viene prodotto dalla Limbic Systems che ne ha lanciate altre due versioni per il Micro BBC e lo Spectrum



sintattici separati, contenuti in riquadri rettangolari.

LIZ DIXON



Schema sintattico di "Begin — Istruzione"

Esaminando il diagramma complessivo di un programma, si vede che le parole Begin e End vengono definite come parte del vocabolario "riservato" del PASCAL e che non richiedono ulteriori diagrammi per spiegare il loro significato. Il vocabolario del PASCAL è composto unicamente dalle 35 parole di significato fisso elencate nel riquadro a fondo pagina. Il programma precedente ne usa solo quattro: Program, Begin, Const (abbreviazione inglese per "costante") e End. La parola che segue Program è un "identificatore" che rappresenta il nome del programma.
Per essere valido, un identificatore deve iniziare con una lettera e contenere solo lettere o numeri. È quindi possibile definire un numero infinito di nomi di programmi, ma non è consentito usare a tale scopo parole riservate. Per esempio:

```
Nome
PASCAL
ProgrammaUno
N
XYZ123
Indirizzo12
PaoloBianchi
IdentificatoreMoltoLungo
```

sono legali, mentre:

```
Prog-1
DIECI%
Franco.Rossi
and
Tempo$Quadrato
Dall'AltoInBasso
300Dati
Terzo valore
```

sono tutti identificatori illegali perché contengono caratteri non alfanumerici o perché iniziano con un numero oppure, è il caso di and, sono parole riservate del PASCAL. L'ultimo esempio non è accettato a causa dello spazio di separazione tra le parole terzo e valore. In PASCAL non esiste differenza di significato tra le lettere maiuscole e quelle minuscole benché in alcune versioni non standard del linguaggio si raccomanda di scrivere le parole riservate in maiuscolo.

Oltre agli spazi ed ai "fine-linea", un altro elemento della sintassi del PASCAL usato come separatore è quello per i "commenti", che possono trovarsi ovunque (eccetto che, ovviamente, all'interno di una parola): i commenti sono delimitati da parentesi graffe.
Passiamo adesso ad esaminare un programma PASCAL leggermente più complesso:

```
Program ProgrammaDue (input, output);
        {Calcola il quadrato di un numero}
Const
        Messaggio = 'Inserire un numero:';
Var
        numero : integer;
Begin
    WriteLn;
    WriteLn;
    write (Messaggio);
    read (numero);
    WriteLn ('Il quadrato di',numero,'è',numero * numero)
End.
```

Gli identificatori input e output abbinati al titolo sono richiesti dal PASCAL e indicano i file esterni o le periferiche con le quali il programma comunica. Di solito, nel caso di un micro, ci si riferisce alla tastiera e allo schermo. L'inclusione di input permette di "leggere" la tastiera tramite l'istruzione read, che è una procedura standard del PASCAL. I parametri di read e di write devono sempre trovarsi tra parentesi tonde.
Il PASCAL usa l'istruzione Var per riservare le aree di memoria destinate a questi parametri. Nel programma appena visto, Var è riferita a un numero intero. Contrariamente al BASIC, che ricorre al simbolo $ per distinguere le variabili stringa da quelle numeriche, il PASCAL possiede una gamma pressoché illimitata di tipi di dati. Poiché è necessario che il compilatore conosca la quantità di memoria riservata per ciascun dato, le dimensioni di ogni variabile devono essere definite tramite un'istruzione Var.
Nel nostro programma, la procedura interna write del PASCAL posiziona il cursore immediatamente dopo il messaggio. La stessa cosa accade con il formato PRINT; del BASIC.
Per passare ad una nuova linea di testo occorre invece usare la procedura WriteLn, dove Ln rappresenta un'abbreviazione della parola Linea.
Può essere utile usare le maiuscole W e L per distinguere meglio le due parole. Un'istruzione WriteLn non seguita da parametri produce semplicemente una nuova linea.
Il PASCAL permette di distinguere i dati variabili da quelli che restano costanti per tutta l'esecuzione del programma. Si noti infatti che mentre la definizione Const è seguita dal segno uguale (=), Var usa i due punti (:).

LE 35 PAROLE RISERVATE DEL PASCAL

And	Case	Do	End	Function	In Label	Not Of	Procedure Program	Set Then	Until Var
Array	Const	Downto	File	Goto	Mod	Or	Record	To	While
Begin	Div	Else	For	If	Nil	Packed	Repeat	Type	With

Il Project della Microsoft

Un programma per i personal IBM e apparecchi compatibili

Sebbene il package *Project* della Microsoft non possieda funzioni grafiche paragonabili a quelle del *MacProject* o del programma *Graphics Manager Environment* della Digital Research, utilizza tuttavia nel modo migliore le limitate capacità grafiche del sistema IBM. Per esempio, abbinando una serie di caratteri "uguale" e "lineetta" al simbolo "maggiore di", si creano frecce del seguente tipo:

= = => ---->

La prima può essere visualizzata in neretto per rappresentare i percorsi "critici", la seconda in colore sfumato per indicare quelli "non critici". I principali fattori da prendere in considerazione, sono i seguenti:

- Qual'è la flessibilità del programma e fino a che punto esso si adatta alle modifiche necessarie via via che il progetto si sviluppa?
- Il programma è in grado di visualizzare con sufficiente chiarezza le correlazioni esistenti tra i vari passaggi, in modo da garantire un'efficiente gestione del progetto?
- Il programma può concorrere in qualche modo alla pianificazione iniziale oppure interviene solo dopo che l'utente ha sviluppato autonomamente il proprio progetto?

In risposta all'ultima domanda si ha che l'utente deve pianificare il progetto su carta, prima di passare alla codifica vera e propria. Infatti il manuale del *Project* afferma che prima di inserire attività, occorre comunicare al pacchetto le seguenti informazioni:

Descrizione di ciascuna attività
Durata dell'attività
Attività che ne precedono l'esecuzione
Data d'inizio

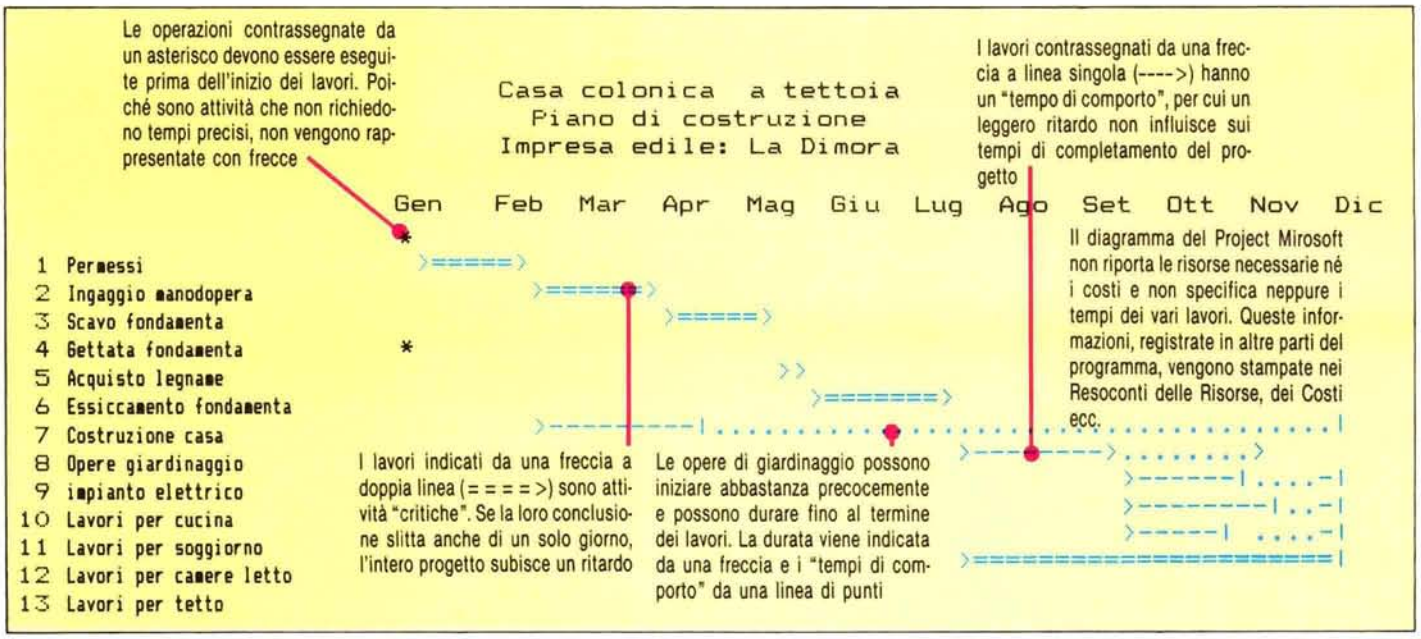
Un'appendice al manuale descrive in 10 pagine queste informazioni preparatorie affermando tra l'altro che il progetto produce i risultati desiderati solo se, inizialmente, vengono definiti con precisione gli obiettivi, i limiti ed i principi di funzionamento. È quindi necessario definire esattamente le diverse operazioni coinvolte nel progetto.
Nessun programma del computer può sopperire ad una definizione inadeguata o imprecisa delle attività trattate, come la pianificazione, l'assegnazione e il controllo delle varie componenti del progetto e anche il bilancio e il controllo dei costi. Una definizione sommaria delle diverse funzioni può creare grossi problemi sin dall'inizio.
Ben pochi fra coloro che possiedono una certa esperienza di tecniche gestionali contesteranno la necessità di questa accurata fase di preparazione. Anche chi considera il computer uno strumento "intelligente", qualitativamente diverso da tutti gli altri, può apprezzare il fatto che il software aiuti a raggiungere i risultati desiderati evitando insufficienze e imprecisioni. Senza dubbio il *Project* svolge questa funzione nel modo migliore, e, grazie alla pianificazione su carta, si possono fare modifiche al progetto anche sostanziali, prima di passarlo al programma.
Subito dopo l'accensione del computer l'utente immette la data e l'ora correnti: queste definiscono il calendario usato dal programma. Tali parametri, comunque, possono essere facilmente modificati anche dopo l'introduzione di molte delle attività previste dal progetto. Se il tempo assegnato a una qualsiasi attività viene modificato, il programma ricalcola immediatamente lo schema delle diverse attività, scambiando in modo appropriato le frecce = = => (passaggi critici) con le frecce ----> (passaggi non critici) e viceversa. Questa rappresentazione istantanea dei cambiamenti avvenuti non solo favorisce la comprensione delle relazioni tra le diverse attività, ma evidenzia efficacemente la nuova configurazione, che potrebbe facilmente sfuggire all'utente. La Microsoft ha già sperimentato questa tecnica sul *Multiplan*, uno dei programmi di lavagna elettronica di maggior successo da quando, nel 1979, venne lanciato il *Visicalc*, il primo degli spreadsheet. In effetti, chi ha una certa dimestichezza con il *Multiplan* impara in breve tempo a usare anche il *Project*, poiché esistono molte affinità tra i due package.

Funzionamento del Project

Dopo una breve comparsa del simbolo della *Microsoft*, il *Project* visualizza lo schermo di lavoro, che appare vuoto se l'utente non ha specificato un nome di file. Sulla sinistra lo schermo presenta una colonna di numeri compresi tra 1 e 19, che indicano le righe, e sul margine superiore la scala dei tempi. Se l'utente ha selezionato in precedenza la data del giorno, la scala inizia da essa, che può essere comunque variata tramite il comando OPTIONS che appare nel menu. Quest'ultimo è visualizzato, sul margine inferiore dello schermo e contiene le seguenti opzioni:

BLANK CALENDAR DELETE EDIT GOTO HELP INSERT MOVE PRINT QUIT RESOURCE SORT TRANSFER XTERNAL

L'opzione in uso (inizialmente ACTIVITY) viene visualizzata sotto il messaggio che invita a selezionare un'opzione oppure a digitare l'iniziale di un comando, mentre nell'angolo inferiore destro compare il nome del progetto da elaborare (se l'utente non lo ha specificato viene assegnato il titolo TEMP).
La scala dei tempi può essere suddivisa in



Progetto edilizio
Questo schema rappresenta la pianificazione per la costruzione di una casa. I tempi di realizzazione appaiono nella parte alta del grafico, mentre sul margine sinistro vengono riportati i lavori in ordine d'esecuzione. La durata di ciascun lavoro è indicata da frecce create con alcuni caratteri della tastiera (< > . = -)

giorni, settimane o mesi ed essere cambiata in qualsiasi momento, per consentire una visione d'insieme dell'intero progetto. Con una scala giornaliera, ad esempio, è possibile visualizzare le date comprese tra il primo di gennaio e il 2 di marzo, mentre una scala mensile consente di osservare un periodo totale di due anni e 3 mesi. Come avviene su una comune lavagna elettronica, i tasti che controllano i movimenti orizzontali del cursore vengono usati per far scorrere lateralmente lo schermo di lavoro ACTIVITY. Tramite l'opzione CALENDAR, l'utente ha la possibilità di evidenziare le festività e gli altri giorni non lavorativi. Il sabato e la domenica vengono normalmente indicati come giorni non lavorativi (com'è logico attendersi da un package anglosassone); questa definizione si può cambiare modificando singolarmente il giorno che interessa. L'utente può visualizzare i mesi successivi con l'istruzione <PAGE DOWN> e quelli precedenti con <PAGE UP>.

Un fatto piuttosto fastidioso, ma comune a tutti gli apparecchi IBM e IBM-compatibili, è che tutte le date vanno inserite secondo il formato americano MM/DD/YY (mese/giorno/anno), per cui 01/07/85, ad esempio, indica il 7 gennaio 1985 e non il 1 luglio 1985.

Dopo aver impostato il calendario, è possibile salvare il file che lo contiene con l'opzione TRANSFER (simile a quella del *Multiplan*). Il programma aggiunge il suffisso .CAL, per distinguere il file da quelli delle attività (.ACT) e da quelli delle risorse (.RES).

A questo punto l'utente introduce le attività che il *Project* deve elaborare e il tempo assegnato a ciascuna di esse, quindi specifica le attività cronologicamente precedenti a quelle immesse, digitando le varie dipendenze separate da virgole. La colonna per i titoli delle attività può contenere al massimo 15 caratteri. I caratteri in eccedenza vengono tuttavia memorizzati e riprodotti in fase di stampa su carta. Dopo l'immissione delle diverse attività, corredate di durata, attività precedenti e data d'inizio, il *Project* visualizza le loro interrelazioni e rappresenta con una serie di punti il "tempo di comporto", ossia il ritardo consentito nell'esecuzione delle attività, senza che questo rallenti il progetto nel suo complesso. Quindi vengono assegnate le risorse alle diverse attività. Dopo l'inserimento di ogni informazione è possibile richiamare la Tabella delle Risorse, con i risultati ottenuti.

Conclusa l'immissione dei dati, l'utente può chiedere una stampa del prospetto completo, una pagina dopo l'altra. Tuttavia, se il computer viene dotato di una scheda grafica ed è collegato a una stampante adeguata (quali la stampante grafica PC IBM o le Epson MX-80, MX-100, FX-80 o FX-100), è possibile ottenere un tabulato generale del progetto, stampato parallelamente alla lunghezza della carta. Questa tecnica permette di avere una visione completa del progetto su di un foglio unico, anziché su diverse pagine di stampa. Inoltre è possibile produrre resoconti dettagliati delle diverse attività, contenenti i tempi (minimi e massimi) per l'avvio e per la conclusione del lavoro, le risorse, i predecessori, i tempi di comporto e una tabella delle attività relative all'intero progetto, che ripropone le stesse informazioni in modo più sintetico.

In realtà, la mancanza di sofisticate funzioni grafiche nel *Project* non è molto limitante e, sebbene si sottolinei più volte la necessità di un'adeguata pianificazione iniziale, è possibile correggere ripetutamente le informazioni e introdurre margini di ritardo inizialmente non previsti.

La velocità del programma è maggiore di quella del *MacProject* per il Macintosh, ma anche il suo prezzo è superiore.

Project Microsoft: Per apparecchi MS-DOS
Editori: Microsoft Inc. (USA)
Autori: MAS, Seattle, USA
Formato: Su disco

DATABASE

Ecco l'ottava tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi

DESCRIZIONE	INDIRIZZO
ON statement routine routine istruzione ON	B915
OPENIN function routine routine funzione OPENIN	BF78
OPENOUT function routine routine funzione OPENOUT	BF7C
OPENUP function routine routine funzione OPENUP	BF80
OR operator routine routine operatore OR	9B3A
OSCLI statement routine routine istruzione OSCLI	BEC2
Op-code base table indirizzo iniziale tabella op-code	84C5
PAGE function routine routine funzione PAGE	AEC0
PAGE statement routine routine istruzione PAGE	9283
PI function routine routine funzione PI	ABCB
PLOT statement routine routine istruzione PLOT	93F1
POINT function routine routine funzione POINT	AB41
POS function routine routine funzione POS	AB6D
PRINT statement routine routine istruzione PRINT	8D9A
PRINT # statement routine routine istruzione PRINT#	8D2B
PROC statement routine routine istruzione PROC	9304
PTR function routine routine funzione PTR	BF47
PTR statement routine routine istruzione PTR	BF30
Pack FAC#1 to &46C onwards compatta FAC#1 da &46C in poi	A385
Pack FAC#1 to &471 onwards compatta FAC#1 da &471 in poi	A37D
Pack FACK#1 to &476 onwards compatta FAC#1 da &476 in poi	A381
Pack FAC#1 to (&4B) compatta FAC#1 da (&4B)	A38D
Parameters; unstacking estrazione di paramentri dallo stack	8CC1
Print A in hexadecimal visualizza A in esadecimale	B545
Print a character visualizza un carattere	B50E
Print a string visualizza una stringa	BFCF

INDIRIZZO	DESCRIZIONE
A907	**ATN function routine** routine funzione ATN
A98D	**COS function routine** routine funzione COS
A998	**SIN function routine** routine funzione SIN
AA91	**EXP function routine** routine funzione EXP
AB12	**Set FAC#1 = FAC#1 ^ A** pone FAC#1 = FAC#1 ^ A
AB33	**ADVAL function routine** routine funzione ADVAL
AB41	**POINT function routine** routine funzione POINT
AB6D	**POS function routine** routine funzione POS
AB76	**VPOS function routine** routine funzione VPOS
AB88	**SGN function routine** routine funzione SGN
ABA8	**LOG function routine** routine funzione LOG
ABB1	**RAD function routine** routine funzione RAD
ABC2	**DEG function routine** routine funzione DEG
ABCB	**PI function routine** routine funzione PI
ABD2	**USR function routine** routine funzione USR
ABE9	**EVAL function routine** routine funzione EVAL
AC2F	**VAL function routine** routine funzione VAL
AC78	**INT function routine** routine funzione INT
AC9E	**ASC function routine** routine funzione ASC
ACAD	**INKEY function routine** routine funzione INKEY
ACB8	**EOF function routine** routine funzione EOF
ACC4	**TRUE function routine** routine funzione TRUE
ACD1	**NOT function routine** routine funzione NOT
ACE2	**INSTR function routine** routine funzione INSTR
AD6A	**ABS function routine** routine funzione ABS
AD8C	**— unary operator** operatore di unità —

L'Apricot Portable

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

Gli apparecchi portatili rappresentano una delle principali aree di sviluppo della microinformatica: accompagnano molti uomini d'affari nei loro viaggi e possono essere adoperati dovunque sia presente un'adeguata presa di alimentazione. Ma si possono usare anche per altri scopi: spesso vengono tenuti in casa e adoperati per terminare lavori iniziati in ufficio su apparecchi compatibili.
Tra le molte case di computer interessate a quest'area del mercato, la ACT (Apricot) è stata una delle ultime a lanciare un portatile compatibile con i propri apparecchi da tavolo. L'Apricot Portable è protetto da una robusta custodia in materiale plastico (simile a una valigetta 24 ore), dotata di un alloggiamento che accoglie l'ampio display a cristalli liquidi. Per aprire la custodia ci sono due agganci (che non sempre funzionano come si deve), posti ai due lati della maniglia.



CHRIS STEVENS

Esame dell'hardware

La tastiera e l'unità centrale sono fissate, con due cinghie, all'interno della custodia che contiene anche due manuali, un microfono e un mouse/trackball. Non deve meravigliare la mancanza di cavi di collegamento per la tastiera e per il mouse poiché le informazioni vengono inviate all'unità centrale tramite un fascio di raggi infrarossi. I dati vengono trasmessi e ricevuti da due piccole cellule fotosensibili, poste sul retro della tastiera e sul davanti del mouse. I segnali generati da questi strumenti vengono raccolti da un "bulbo" di messa a fuoco situato sul computer. L'utente può tenere la tastiera sulle proprie ginocchia e appoggiare il computer su di un tavolo, anche a una certa distanza.
Questa tecnica a raggi infrarossi presenta però due inconvenienti. Innanzi tutto, se più portatili Apricot vengono fatti funzionare contemporaneamente all'interno di una stanza, si possono verificare interferenze tra i segnali generati dagli apparecchi. Per eliminare questa anomalia, la Apricot ha prodotto un cavo a fibre ottiche che collega fisicamente la tastiera all'unità centrale.
Un secondo problema è legato all'uso del mouse: se l'utente posiziona la tastiera di fronte al computer, questa spesso ostacola la ricezione dei segnali provenienti dal mouse, agendo come schermo ai segnali diretti al bulbo, che si trova appena sopra la superficie del tavolo. D'altra parte, se la tastiera viene spostata lateralmente, l'unità centrale può non ricevere i segnali che da essa provengono.

Uno dei problemi, nella progettazione di portatili derivati da apparecchi da tavolo, è la compressione delle funzioni del computer in spazi più limitati, soprattutto per quanto riguarda la tastiera. I fabbricanti sono ricorsi alle tecniche più sofisticate per ottenere tastiere più piccole e leggere, pur dotate di tutte le funzioni occorrenti, ma spesso è necessario qualche compromesso. In questo apparecchio, la versione portatile della tastiera, derivata da quella standard dell'Apricot, manca infatti del display LCD per la calcolatrice e gli 87 tasti sono più ravvicinati. La sua configurazione è identica a quella dell'Apricot da tavolo, fatta eccezione per i tasti funzione, che qui sono sul lato destro della tastiera.
Per compensare almeno in parte l'eliminazione del display LCD, che nella versione da tavolo era controllato con comandi a singolo tasto, l'Apricot ha dotato il portatile di un nuovo tasto funzione e di un tasto per visualizzare l'ora e la data. I tasti QWERTY, posti a sinistra, sono fiancheggiati su entrambi i lati da tasti di controllo; in particolare, quelli per il cursore si trovano in basso sulla destra. Nel mezzo è stata inserita la tastierina numerica della calcolatrice evidenziata per il colore grigio pallido; la tastiera ha un aspetto elegante, ma l'eccessiva vicinanza tra i tasti la rende poco funzionale.

Un portatile futuristico
Nell'Apricot Portable è stata impiegata una tecnologia molto avanzata. L'apparecchio possiede un display di 80 x 25 caratteri, che permette di visualizzare la maggior parte dei programmi applicativi sotto MS-DOS. La tastiera e il mouse/trackball mostrati in figura non vengono collegati al computer in modo convenzionale, bensì ricorrono a un fascio di raggi infrarossi per inviare i propri segnali. Sulla destra dello schermo c'è il microfono, che il computer usa abbinato a un apposito programma di riconoscimento vocale

Invece dei convenzionali tasti "scavati", l'Apricot Portable è dotato di una tastiera incassata, simile a quella del QL. Poiché i tasti non sporgono dal corpo principale, la tastiera è più compatta ma anche più difficile da usare: spesso accade di premere contemporaneamente due tasti vicini. Questa eccessiva compattezza è aggravata dall'assenza di separazioni tra i tasti funzione quelli alfanumerici e quelli della calcolatrice. In compenso le dimensioni dei tasti Shift, Stop e Caps Lock sono doppie rispetto a quelle dell'Apricot standard facilitandone l'individuazione.
Sopra la tastiera, il display della calcolatrice è stato sostituito da quattro pulsanti, che abilitano altrettante funzioni extra. Il pulsante Reset provoca il ricaricamento del dischetto inserito nell'apposita unità; il secondo tasto permette di modificare la velocità di ripetizione dei tasti; il terzo chiama una routine che azzera l'orologio interno del computer e il quarto, infine, disabilita la tastiera. Quest'ultima funzione, utile quando si lavora con il mouse o durante l'immissione del parlato: impedisce che l'involontaria pressione di un tasto alteri le informazioni ricevente il computer.
Lo schermo a cristalli liquidi incorporato è fissato al corpo principale del computer. La sua definizione è minore rispetto a quella di altri apparecchi simili, mentre la colorazione olivastra dello sfondo è più che accettabile. Il microfono alla destra dello schermo viene collegato, con un cavetto a spirale, a una presa sul fianco sinistro del computer.

Riconoscimento del parlato

Forse la caratteristica più originale dell'Apricot Portable è il programma di riconoscimento vocale, in grado di identificare comandi vocali, interpretarli come istruzioni e eseguirli nel modo consueto. Nel modo "Voice Driven Application", la macchina accetta e riconosce fino a 63 parole di comando per qualsiasi programma applicativo, quali il *WordStar* e il *SuperCalc*. Un programma di apprendimento vocale chiede all'utente se la voce che impartirà i comandi è maschile, femminile o infantile e se l'ambiente in cui il portatile opera è rumoroso. In genere, dopo aver ripetuto più volte uno stesso comando, il computer lo "impara" ed è capace di identificarlo indipendentemente dalle variazioni di intonazione. Purtroppo talvolta il programma fraintende i comandi o li ignora. Perciò è sconsigliabile usare comandi vocali per DELETE e FORMAT.
Come gli altri apparecchi della gamma Apricot, anche il Portable usa dischetti Sony da 3 pollici, che vengono inseriti nell'unica unità a dischetti posta sul lato destro. Sul retro del computer ci sono tre interfacce, protette da un pannello in plastica: una porta parallela Centronics per stampanti, una presa RS232 per modem esterno o altre periferiche seriali e una porta per joystick di tipo Atari. Questa interfaccia non è dedicata ai programmi di gioco, ma serve per il collegamento diretto di un mouse o di una trackball, quando le circostanze sconsigliano l'uso dei raggi infrarossi.
Il mouse a raggi infrarossi, di corredo, è alimentato da una pila. L'utente può adoperarlo sul tavolo di lavoro come un comune mouse oppure, se lo rovescia, può tenerlo in mano azionando la sua sfera di controllo o, ancora, appoggiarlo sul tavolo e usarlo come una trackball; la prima tecnica si dimostra migliore della seconda, in quanto l'impiego come trackball spesso porta l'utente ad allontanare troppo il dispositivo.
Nonostante la difficoltà di lettura dello schermo e la scarsa affidabilità delle unità d'ingresso, l'Apricot Portable resta una macchina interessante e l'ACT ha fatto notevoli sforzi per dotarlo delle tecnologie più avanzate. Tuttavia la maggior parte degli utenti preferisce a queste tecniche futuristiche, affidare l'elaborazione dei propri programmi ad apparecchi più semplici e più a lungo sperimentati.

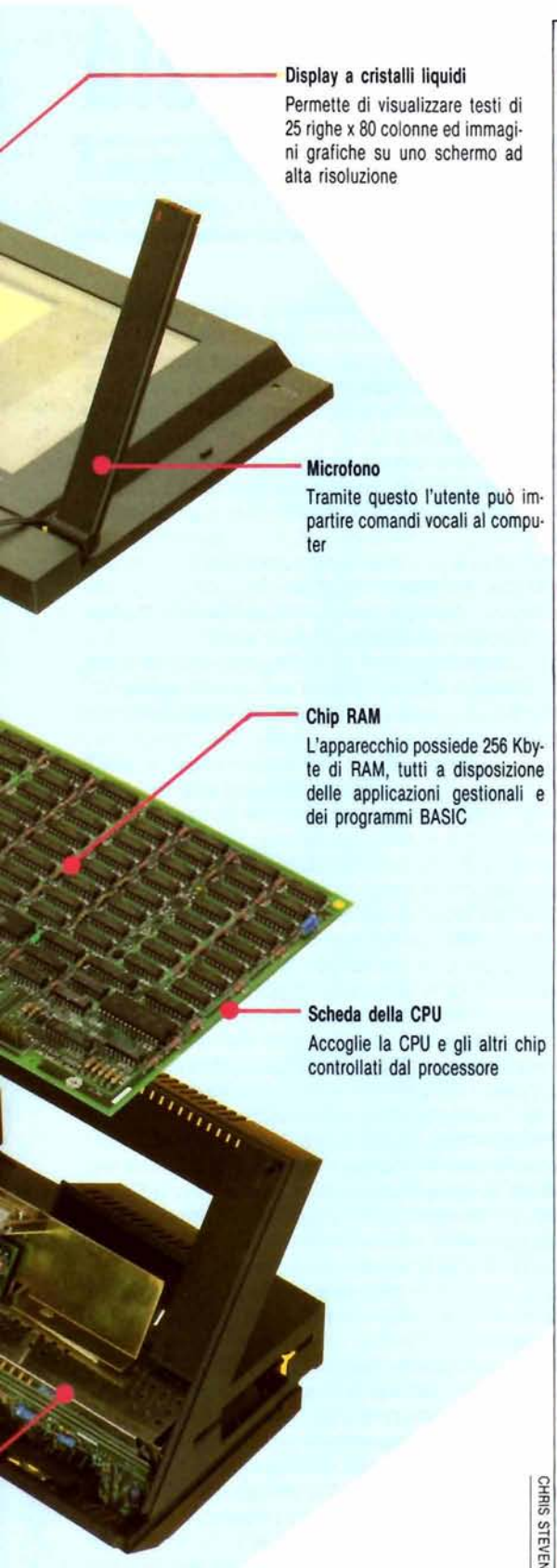


CPU
L'apparecchio usa come unità centrale d'elaborazione il processore 8086 a 16-bit della Intel

ROM d'interfaccia
Poiché l'Apricot Portable è un computer a 16 bit, questi chip trasmettono alla scheda d'interfaccia blocchi di byte in formato basso/alto

Scheda d'interfaccia
Contiene i chip che controllano il funzionamento delle unità di I/O (stampanti, unità a dischetti, ecc.)

Unità a dischetti
Questa unità usa dischetti da 3 pollici a doppia faccia e doppia densità, compatibili con gli altri apparecchi della stessa Casa



Display a cristalli liquidi

Permette di visualizzare testi di 25 righe x 80 colonne ed immagini grafiche su uno schermo ad alta risoluzione

Microfono

Tramite questo l'utente può impartire comandi vocali al computer

Chip RAM

L'apparecchio possiede 256 Kbyte di RAM, tutti a disposizione delle applicazioni gestionali e dei programmi BASIC

Scheda della CPU

Accoglie la CPU e gli altri chip controllati dal processore

CHRIS STEVENS



Software dell'Apricot

L'Apricot Portable è corredato da dischetti Sony da 3 pollici, contenenti una serie di pacchetti applicativi. Il "Supercalc" (lavagna elettronica), il "SuperPlanner" (schedario e calendario) e il "SuperWriter" (word processor) sono solo alcuni dei programmi forniti dalla Casa



Trackball a raggi infrarossi

Ruotando la sfera, è possibile provocare movimenti corrispondenti del cursore a freccia visualizzato sullo schermo. Quando il cursore viene posizionato su un'icona, basta premere uno dei due tasti ai lati della trackball per selezionare l'applicazione indicata. Se si rovescia lo strumento e lo si appoggia sul tavolo con la sfera a contatto della superficie, esso si comporta come un mouse



Porte d'interfaccia

Le interfacce si trovano sul retro della macchina, sotto un pannello di materiale plastico. Il connettore inferiore è una porta parallela di tipo Centronics, sormontata da una presa RS232, usata per collegare il computer a un modem esterno. Sull'interfaccia di sinistra possono essere innestati un mouse di tipo tradizionale o una trackball

APRICOT PORTABLE

DIMENSIONI

mm. 450 x 335 x 200

CPU

8086 della Intel

MEMORIA

256 Kbyte di RAM (211 Kbyte liberi per i programmi applicativi) e 32 Kbyte di ROM

VIDEO

Elettroluminescente a cristalli liquidi, con una risoluzione testuale di 80 x 25 caratteri e grafica di 640 x 256 pixel. Collegando il computer ad un monitor esterno, sono disponibili 16 colori

INTERFACCE

RS232 seriale, parallela Centronics e un connettore per mouse. È previsto anche un collegamento ottico per il mouse o la tastiera e una presa jack per il microfono

UNITÀ A DISCHETTI

Il computer è dotato di un'unità a dischetti di 3 pollici, in doppia faccia e doppia densità, per un totale di 720 Kbyte

SISTEMI OPERATIVI

MS-DOS, CP/M-86, CP/M concorrente

TASTIERA

Compatibile con quella del personal IBM, possiede 87 tasti di cui 10 di funzione e quattro pulsanti con funzioni extra

DOCUMENTAZIONE

Un manuale per "Partire con il sistema" descrive l'impiego del computer e delle funzioni associate, compreso il programma per il riconoscimento vocale. Un secondo manuale fornisce spiegazioni sui programmi applicativi in dotazione ("SuperCalc", "SuperPlanner" e "SuperWriter")

PRO

Questo apparecchio è senza dubbio all'avanguardia per quanto concerne la tecnologia applicata nel suo progetto, particolarmente per il tentativo di dotare il sistema di un modulo di riconoscimento vocale

CONTRO

L'avanzata tecnologia del progetto costituisce un'arma a doppio taglio, non essendo ancora sufficientemente sperimentata e collaudata. Il display non è sempre ben leggibile

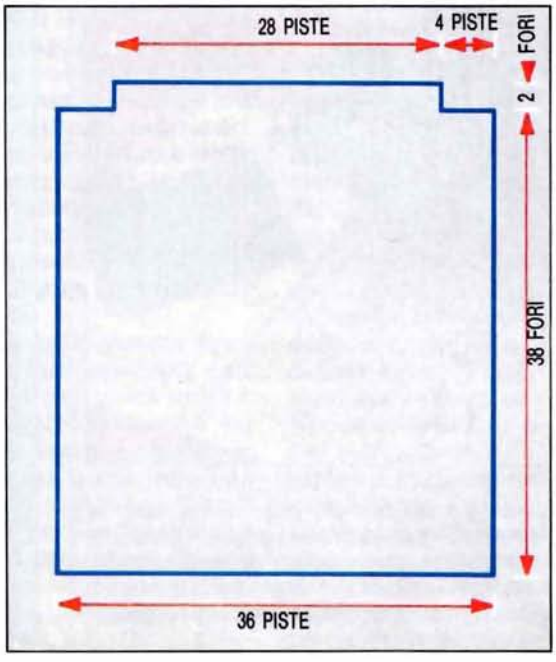
Realizziamo l'interfaccia

I dettagli costruttivi del circuito che consente allo Spectrum di pilotare il robot

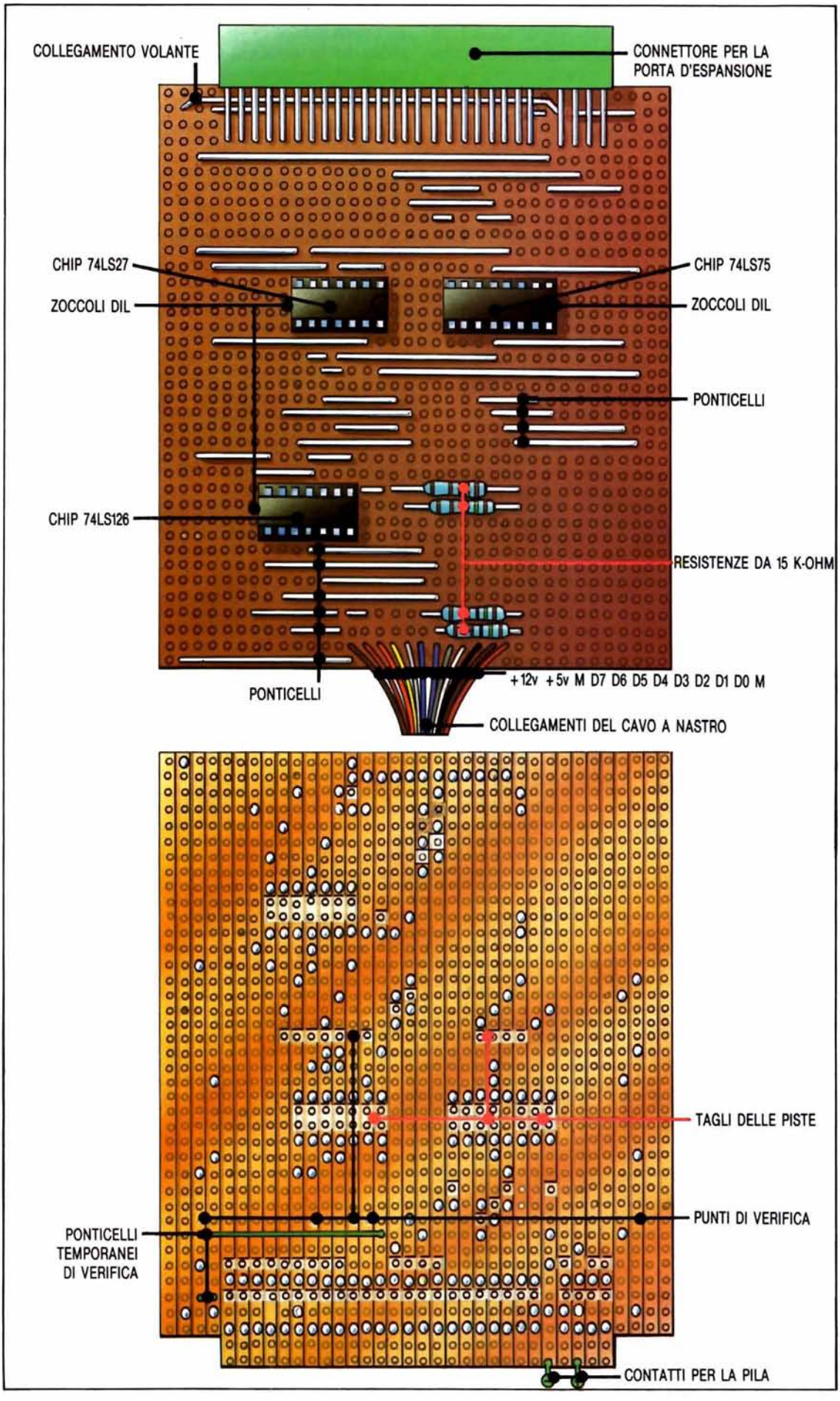
Il quinto contatto da sinistra del connettore a pettine progettato per lo Spectrum deve essere ostruito. La prima operazione da compiere è inserire un "blocco" nel connettore a pettine della porta d'espansione per lo Spectrum. Questo costringe ad innestare l'interfaccia nella presa con la giusta orientazione.
Dopo aver sagomato una piastrina tipo veroboard di forma adeguata (vedere figura), si usi uno degli angoli asportati per realizzare il blocco per il contatto. Una volta eliminato il rivestimento di rame, si inserisca il rettangolino di veroboard nel quinto contatto del connettore a pettine, fissandolo con della colla, e si taglino i piedini di collegamento. Quindi occorre tagliare le piste della scheda secondo lo schema, facendo attenzione che ogni pista venga interrotta per l'intera larghezza. Eventuali contatti residui impediscono il funzionamento dell'interfaccia e potrebbero danneggiare lo Spectrum.
Passiamo quindi ad assemblare i vari componenti del circuito sulla scheda, iniziando dai ponticelli, la cui lunghezza deve presentare margini di riserva. Si saldi un'estremità del ponticello e poi lo si tenda prima di saldare la seconda; infine si taglino le estremità che sporgono dalla scheda. Dopo aver ripetuto questa operazione per tutti i ponticelli, si saldi il conduttore per il collegamento volante, utilizzando uno dei fili di un cavo a nastro.
Terminata questa operazione, si saldi il connettore a pettine sulla scheda. Poiché il sistema di collegamento del connettore è stato progettato per schede a doppia faccia (come quelle dello Spectrum), mentre la piastrina di veroboard è a singola faccia, l'operazione di fissaggio è piuttosto complicata. Per prima cosa occorre piegare a 90° una delle due serie di piedini: questi costituiscono la linea inferiore dei piedini, quando il connettore viene posizionato sulla scheda. Poi si salda un filo, lungo circa 3 cm., su ciascun piedino della linea superiore. I piedini piegati vanno inseriti nella terza riga di fori a partire dal margine del connettore e, dopo aver piegato ad angolo retto anche i fili provenienti dalla linea superiore, si inseriscano questi nella sesta riga di fori. Adesso si spalmi uno strato di colla sulla parte della scheda tra i due angoli tagliati e si fissi su di essa il connettore a pettine. Infine si saldino le due linee di piedini alle piste di rame facendo attenzione che le saldature non creino contatti tra le piste. Per maggior sicurezza, conviene passare tra le piste una lama di coltello, per asportare qualsiasi sbavatura di stagno.
A questo punto si saldino sulla scheda gli altri componenti. Prima si fissino i tre zoccoli DIL, poi le 4 resistenze da 15 K-ohm e quindi i tre chip, facendo attenzione a disporli in modo corretto. Le tacche di riferimento devono essere tutte orientate nella stessa direzione, ossia verso destra, quando il connettore dell'interfaccia è rivolto verso l'alto.
A questo punto l'interfaccia è completa ma, prima di innestarla sullo Spectrum, occorre collaudarla. Poiché la porta d'espansione dello Spectrum è collegata non solo ai bus degli indirizzi e dei dati e al processore, ma anche alle linee di alimentazione a 12 e 9 volt, è indispensabile controllare bene che le uscite di tensione non vengano collegate ai delicati

ELENCO DEI COMPONENTI

1	Chip 74LS27
1	Chip 74LS75
1	Chip 74LS126
4	Resistenze da 15 K-ohm, 1/3 watt
1	Bobina di filo stagnato
2	Zoccoli DIL a 14 piedini
1	Zoccolo DIL a 16 piedini
5 m.	Cavo a nastro a 20 linee
1	Connettore a pettine 2x28 linee
1	Connettore D a 15 linee
1	Cappuccio per connettore D a 15 linee
1	Piastrina tipo veroboard (50 fori x 36 piste)



Sagomatura della scheda
I componenti dell'interfaccia per lo Spectrum vengono assemblati su una piastrina di tipo veroboard di queste dimensioni. Uno dei due angoli asportati viene usato come "blocco" da inserire nel connettore d'espansione della porta. La parte sporgente della scheda serve per il montaggio del connettore i cui piedini vengono saldati alle piste di rame



SCHEMA DEL CIRCUITO

Si saldino i ponticelli e le resistenze sulla faccia del veroboard destinata ai componenti, quindi si fissi il connettore per la porta d'espansione e gli zoccoli DIL. Verificare il corretto orientamento dei tre chip: le tacche devono essere rivolte verso destra.

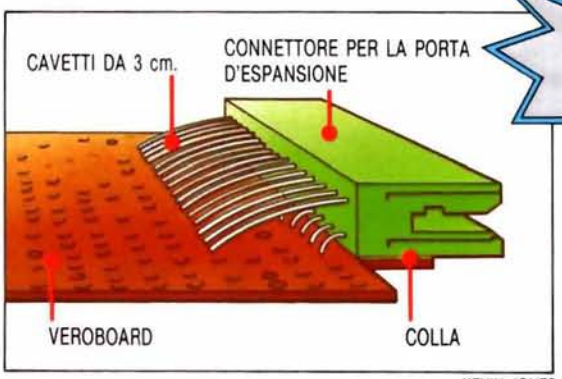
Si controlli, sul lato delle piste ramate (magari usando un tester), che tutti i tagli delle piste interrompano effettivamente eventuali contatti elettrici. I ponticelli in verde sono collegamenti temporanei, usati per collaudare il circuito e vanno rimossi al termine della verifica. Infine si ricontrollino ancora una volta tutte le saldature e si confronti il circuito finito con lo schema base, prima di innestare l'interfaccia sullo Spectrum

circuiti elettronici, che funzionano a livelli elettrici TTL (Transistor Transistor Logic). Si ricontrolli l'intera scheda, verificando che non ci siano contatti tra le piste, e che la disposizione dei ponticelli e di tutti i componenti corrisponda allo schema del circuito e i chip siano correttamente orientati.

Nella prossima parte di LABORATORIO viene realizzato il cavo a nastro per collegare il robot all'interfaccia e vengono sviluppate le versioni Spectrum di alcuni programmi presentati in precedenza.

Collegamenti del connettore

I piedini del connettore a pettine, usato sulla scheda d'interfaccia, devono essere leggermente modificati. Quelli del quinto contatto da sinistra vanno eliminati, si pieghino poi ad angolo retto quelli della linea inferiore e si saldi a ciascun piedino della linea superiore un cavetto di circa 3 cm., per poi inserire i piedini inferiori nella terza riga di fori a partire dal bordo e quelli superiori nella sesta riga. Infine, prima di effettuare le saldature, si incolli il connettore sulla scheda



KEVIN JONES

ATTENZIONE!

La porta d'espansione dello Spectrum è collegata ai delicati circuiti elettronici dell'apparecchio e presenta anche "uscite" sotto tensione (a 12 e 9 volt) che, se collegate per errore al bus degli indirizzi o a quello dei dati, possono danneggiare gravemente il computer. Verificare con estrema attenzione TUTTI i collegamenti prima di innestare l'interfaccia nella porta d'espansione e accendere il sistema

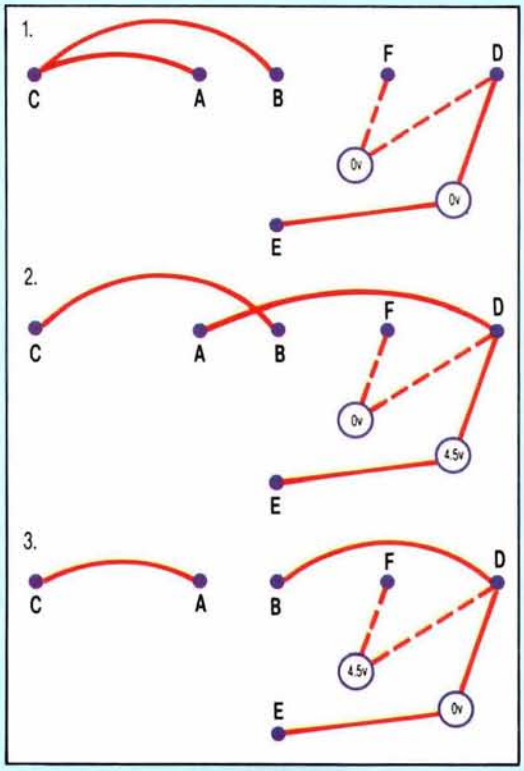
COLLAUDO DELLA SCHEDA

Prima di innestare l'interfaccia sullo Spectrum occorre eseguire una serie di verifiche per accertarsi del corretto funzionamento della logica di decodifica. Gli strumenti necessari sono: un multimetro, una pila da 4,5 volt e una serie di cavetti rivestiti. Per prima cosa si saldino sulla faccia del veroboard che contiene le piste i due ponticelli verdi, visibili nello schema del circuito, e si colleghino i poli della pila ai due appositi contatti sulla scheda. Quindi si colleghino, con due cavetti rivestiti, i punti A e B al punto C, controllando i livelli di tensione nei punti E e F. Tenendo il puntale negativo del misuratore sul punto D, si tocchi con quello positivo prima il punto E, poi il punto F. Si dovrebbero ottenere "letture" comprese tra 1 e 0 volt.

Si dissaldi l'estremità del cavetto che collega C ad A, per saldarla sul punto D. Appoggiando il puntale negativo su D, si tocchino nuovamente E ed F. Le rispettive letture dovrebbero essere 4,5 e 0 volt.

Adesso si dissaldino le estremità dei cavetti che collegano B a C ed A a D per saldarle, rispettivamente, su A e su B: i due cavetti adesso collegano i punti C-A e B-D. Le nuove letture di E ed F dovrebbero essere 4,5 e 0 volt. Se tutte le letture eseguite su E ed F corrispondono ai valori detti, possiamo disconnettere la pila, i due ponticelli verdi e i due cavetti rivestiti. Se invece qualche lettura non dovesse corrispondere, occorre ricontrollare con estrema attenzione tutta la scheda. Finché le letture non risultano corrette, NON SI DEVE ASOLUTAMENTE INNESTARE l'interfaccia sullo Spectrum! La tabella a fianco mostra i vari collegamenti di collaudo e le rispettive letture corrette

COLLEGAMENTI	LETTURE
A-C, B-C	E = 0v F = 0v
A-D, B-C	E = 4,5v F = 0v
A-C, B-D	E = 0v F = 4,5v



Questo schema illustra i diversi collegamenti di collaudo tra i punti A, B, C, e D e le letture dei corretti voltaggi ottenute toccando i punti E ed F nelle tre fasi di verifica

Ancora sui file del BBC

L'ultima parte dedicata alle routine ROM per la gestione dei file sul Micro BBC

La routine OSFIND, il cui indirizzo è &FFCE, permette di aprire un file e di registrarvi i byte uno alla volta. Equivale quindi all'apertura di un file in BASIC con i comandi OPENOUT, OPENIN o OPENUP. Inoltre permette di chiudere i file, operando come l'istruzione CLOSE# del BASIC. Il registro A indica quale operazione è richiesta, mentre i registri X e Y puntano al nome del file in memoria.

- *A=0:* indica che il file deve essere chiuso. In questo caso il registro Y contiene il numero assegnato al file in fase di apertura. Ponendo questo valore a 0, otteniamo la chiusura di tutti i file aperti.
- *A=& 40:* equivale all'istruzione OPENIN del BASIC. Il file viene aperto solo "in lettura" per estrarre dati (byte).
- *A=& 80:* corrisponde al comando OPENOUT del BASIC. Il file viene aperto solo "in scrittura".
- *A=& CO:* equivale a OPENUP del BASIC. Il file viene aperto sia in lettura che in scrittura. Il suo impiego riguarda soprattutto i sistemi che gestiscono file ad accesso casuale.

Tuttavia, la chiamata OSFIND per aprire un file può non produrre il risultato desiderato, se il file su disco è, ad esempio, protetto "in scrittura". In tal caso il file non può essere aperto e, al ritorno dalla chiamata, il registro A risulta posto a 0. Se invece il file viene aperto, A riporta il numero del file che va salvato.

Le seguenti chiamate ROM del sistema d'archiviazione hanno un aspetto comune: usano il numero del file generato da OSFIND.

La chiamata OSARGS

Questa routine (con indirizzo &FFDA) svolge diversi compiti. Una sua particolarità è che il blocco dei parametri, composto solo da 4 byte, viene memorizzato nella pagina zero. I registri X ed Y contengono rispettivamente l'indirizzo del primo byte del blocco e il numero del file, mentre A indica l'operazione da eseguire. Tuttavia, in una particolare chiamata a OSARGS, il registro Y viene posto a 0. In questo caso la routine non riporta il numero del file, ma fornisce invece informazioni sul sistema d'archiviazione in uso oppure comunica che tutti i file aperti sul sistema d'archiviazione sono stati aggiornati in base alle informazioni eventualmente contenute nei buffer per i file. Esaminiamo queste due routine:

- *Y=0: A=0:* una chiamata a OSARGS con questi due parametri e con il registro X puntato su un'area vuota della pagina zero fornisce informazioni sul sistema d'archiviazione in uso. L'informazione viene comunicata dal registro A. La chiamata OSARGS è l'unica in grado di lavorare su un sistema d'archiviazione a nastro. La tabella a destra permette di interpretare i valori riportati dal registro A.
- *Y=0: A=255:* questa routine si differenzia dalla precedente perché A contiene il valore 255. Una chiamata a OSARGS con questi parametri porta all'aggiornamento dei file aperti in base ai dati contenuti nei buffer del computer. Se tutti i file sono chiusi, l'aggiornamento non ha luogo.
- *Y=Numero del file:* se Y contiene il numero del file, il tipo d'operazione eseguita dipende dal contenuto del registro A (vedere la tabella seguente). Il "puntatore sequenziale" di un file aperto comunica al sistema operativo la locazione da dove verrà letto o in cui verrà scritto il byte successivo. Il puntatore viene quindi aggiornato da queste routine oppure dalla funzione PTR# del BASIC. Come è logico aspettarsi, l'informazione viene memorizzata nel blocco dei parametri in pagina 0 con il suo byte meno significativo (BMS) all'indirizzo (X+0). L'informazione estratta dal sistema d'archiviazione per mezzo di OSARGS viene depositata nel blocco dei parametri per un successivo impiego.

Contenuto del registro A	Descrizione
0	Legge la posizione corrente del puntatore sequenziale sul file indicato nel registro Y
1	Copia nel puntatore sequenziale il valore contenuto nel blocco dei parametri. Simile alla funzione PTR#n del BASIC
2	Inserisce la lunghezza del file nel blocco dei parametri
255	Aggiorna il file (il cui numero è registrato in Y) in base al contenuto dei buffer

Dalla descrizione di questa chiamata risulta evidente perché il blocco dei parametri di OSARGS deve trovarsi alla pagina zero: per passare l'indirizzo si dispone solo del registro X.

Contenuto del registro A	Descrizione
0	Nessun sistema d'archiviazione
1	Nastro (1200 baud)
2	Nastro (300 baud)
3	ROM
4	DISCO
5	Sistema in rete
6	Telesoft

OSBGET e OSBPUT

Queste routine vengono usate, rispettivamente, per leggere un singolo byte da un file aperto e per scrivere un singolo byte in un file aperto. OSBGET viene chiamata all'indirizzo &FFD7 ed è facile da usare: nel registro Y viene caricato il numero del file fornito da OSFIND e la lettura di un byte avviene chiamando &FFD7. Il byte viene letto dalla posizione corrente del puntatore sequenziale e depositato in A. OSBPUT esegue l'operazione inversa: scrive un byte nella posi-

zione corrente indicata dal puntatore sequenziale all'interno del file. Per chiamare OSBPUT, il byte da scrivere sul file viene depositato in A, mentre in Y va caricato il numero del file. Una chiamata a &FFD4 consente di accedere a OSBPUT. La seguente routine mostra come usare OSBPUT per scrivere un singolo byte in un file.

```
LDA   #&80 /si prepara ad aprire un file
LDX   #nome MOD 256
LDY   #nome DIV 256
JSR   OSFIND    /apre il file
STA   &71       /conserva il numero del file
TAY             /lo deposita anche nel registro Y
LDA   #65       /il byte da scrivere
JSR   OSBPUT    /esegue la scrittura
LDA   #0
LDY   &71       /si prepara a chiudere il file
JSR   OSFIND    /chiude il file
RTS             /fine
```

La chiamata OSGBPB

Questa routine, il cui indirizzo è &FFD1, consente di trasferire blocchi di byte dalla memoria ai file aperti e viceversa. Inoltre fornisce informazioni sul sistema d'archiviazione su disco, se questo è in uso, ma non sul sistema d'archiviazione su nastro. Questo fatto non deve sorprendere, in quanto OSGBPB si dimostra veramente utile solo quando vengono impiegati file ad accesso casuale con programmi in codice macchina.
La seguente tabella illustra il blocco dei parametri che, come di consueto, viene indirizzato dal contenuto dei registri X e Y. Il registro A indica l'operazione da eseguire.

Blocco dei parametri di OSGBPB	
Indirizzo	**Descrizione**
0	Numero del file, fornito da OSFIND
1	BMS dell'indirizzo del dato da leggere o scrivere
2	Non specificato
3	Non specificato
4	BPS dell'indirizzo del dato
5	BMS del num. di byte da estrarre o inserire nel file
6	Non specificato
7	Non specificato
8	BPS del num. di byte da spostare
9	BMS della posizione del puntatore sequenziale da usare. (È quindi possibile scegliere il punto di scrittura o di lettura all'interno del file)
10	Non specificato
11	Non specificato
12	BPS della posizione del puntatore sequenziale

BMS = Byte Meno Significativo
BPS = Byte Più Significativo

Se si sta usando un modello standard del Micro BBC, l'indirizzo è indicato dal contenuto dei byte 1 e 4 del blocco dei parametri. La chiamata consente 4 diverse operazioni di trasferimento di dati: due di lettura e due di scrittura. Una operazione di lettura e una di scrittura accedono all'informazione contenuta nella locazione del file indicata dal puntatore sequenziale, registrato all'interno del blocco dei parametri. Le altre due operazioni di lettura e scrittura ignorano completamente tale informazione e adoperano il puntatore sequenziale qualunque sia la posizione indicata all'interno del file. Quindi A=1 scrive i byte sul puntatore specificato all'interno del blocco; A=2 li scrive sul file ignorando il dato del blocco dei parametri; A=3 legge i dati dal puntatore sequenziale indicato dal blocco dei parametri mentre A=4 li legge ignorando il blocco dei parametri. Il seguente programma usa le chiamate OSGBPB per scrivere e leggere un semplice file di dati:

```
10 REM solo per sistema a dischi
20 DIM blocco 20
30 DIM dati 100
40 $dati = "Questo è un programma di prova!"
50 blocco!1 = dati: REM indirizzo dei dati da scrivere
60 blocco!5 = 100: REM quantità di dati da scrivere
70 blocco!9 = 0: REM puntatore sequenziale = 0
80 PRINT "Apertura del file in uscita"
90 Y% = OPENOUT ("FILE"): REM apre il file
100 blocco?0 = Y%: REM numero del file
110 X% = blocco MOD 256
120 Y% = blocco DIV 256
130 A% = 1: REM prepara la scrittura
140 CALL &FFD1: REM esegue la scrittura
150 CLOSE £ (blocco?0)
160 PRINT "Chiusura del file"
170 TIME = 0: REPEAT: UNTIL TIME = 400
180 PRINT "Apertura del file in ingresso"
190 PRINT "Lettura dei dati"
200 $dati = STRING$(100," "):REM cancella l'area per i dati
210 Y% = OPENIN ("FILE"): REM apre il file
220 blocco?0 = Y%: REM numero del file
230 blocco!1 = dati: REM indirizzo del dato per l'operazione di lettura
240 blocco!5 = 100: REM numero dei byte
250 blocco?9 = 0: REM puntatore sequenziale a 0
260 X% = blocco MOD 256
270 Y% = blocco DIV 256
280 A% = 3: REM prepara la lettura
290 CALL &FFD1: REM esegue la lettura
300 CLOSE £ (blocco?0)
310 PRINT "Chiusura del file"
320 PRINT $dati
330 END
```

Lo stato del flag di riporto (carry), dopo la chiamata a $FFD1, indica se il trasferimento ha avuto successo o meno. Se C contiene il valore 0, il trasferimento è avvenuto in modo corretto; se C vale 1, il trasferimento è fallito.
L'ultima chiamata al SO, dedicata alle operazioni d'archiviazione, è OSFSC, usata molto raramente nella programmazione. Questa chiamata si differenzia dalle precedenti perché non possiede un indirizzo di chiamata diretto: viene quindi chiamata all'indirizzo contenuto nel suo vettore. Nei programmi in codice macchina, questa chiamata viene effettuata con l'istruzione:

```
JMP (&21E)
```

Esistono altre routine del SO che svolgono specifiche funzioni per particolari sistemi d'archiviazione. Ad esempio, le chiamate OSWORD vengono impiegate per accedere al Chip di Controllo dei Floppy Disk nei sistemi d'archiviazione su disco, mentre un paio di chiamate OSBYTE eseguono alcune semplici mansioni nel sistema d'archiviazione.
Termina qui la nostra panoramica sul ruolo del sistema operativo del Micro BBC nelle operazioni d'archiviazione dell'apparecchio.

Digitaya

In quest'ultima parte viene presentato il listato completo del gioco d'avventura

La struttura di *Digitaya* ha molte affinità con quella di *Foresta Incantata*, ma il gioco si sviluppa in uno scenario molto più vasto.
La mappa iniziale di *Digitaya* è composta da 100 locazioni, che corrispondono ad altrettanti elementi interni di un computer (memoria, bus dati, processore e via dicendo). L'obiettivo del giocatore è quello di ritrovare il misterioso Digitaya, imprigionato in qualche recesso della macchina. Durante la ricerca, il giocatore deve affrontare molti pericoli, come la porta per il joystick, la tabella dei vettori e i "bug" disseminati nel programma, affidandosi alle proprie conoscenze della struttura interna del computer, per liberare Digitaya.
Il programma *Digitaya* usa routine simili a quelle di *Foresta Incantata* per eseguire le operazioni fondamentali, come spostarsi da una locazione ad un'altra, prendere e posare gli oggetti e "guardarsi" attorno. Tuttavia alla sua struttura di base sono state aggiunte alcune routine speciali, che introducono nel gioco nuovi pericoli e trabocchetti.
Prima di iniziare a giocare è consigliabile progettare sulla carta il percorso che si intende seguire all'interno dello scenario di gioco. Per poter vincere, il giocatore deve innanzi tutto tentare di capire la logica seguita dal creatore del programma. Un altro utile sistema è quello di ricostruire sulla carta la mappa delle locazioni usate. *Digitaya* è stato progettato su una griglia bidimensionale, ma non tutte le mappe dei giochi d'avventura sono a due dimensioni. È infatti possibile realizzare mappe tridimensionali pur attenendosi alla struttura di programma descritta per *Digitaya*. Dovrebbe essere possibile sviluppare anche giochi quadrimensionali, usando il tempo come quarta dimensione. Un simile gioco, basato su una mappa in continua trasformazione, senza dubbio sconfiggerebbe anche i più smaliziati e geniali appassionati di giochi d'avventura.
Il listato presentato nelle pagine seguenti è per il Commodore 64, ma funziona anche su molti degli apparecchi dotati di un BASIC tipo Microsoft. Le annotazioni fornite riguardano la versione del gioco per il Micro BBC.
L'eliminazione di alcune istruzioni REM consente di alleggerire il lavoro di battitura del listato. Tuttavia, per avere la certezza che il programma funzioni in modo corretto, è necessario conservare tali linee, anche se composte solo da istruzioni REM, poiché i numeri di linea di queste vengono usati nelle chiamate GOSUB. La loro eliminazione può dar luogo ad un errore UNDEFINED STATEMENT.

Basic & Basic

Spectrum:

Lo Spectrum usa un sistema tutto suo per trattare le variabili e le matrici stringa, quindi la versione "Digitaya" per questo apparecchio richiede numerose modifiche, peraltro già descritte in precedenza. Tuttavia, per facilitare la correzione del listato, riproponiamo qui le principali modifiche da introdurre. Innanzi tutto lo Spectrum accetta solo nomi di variabili stringa a singolo carattere. Tutte le conversioni necessarie per i nomi di variabili sono riportate nella tabella a pag. 1477.
Inoltre lo Spectrum riconosce solo matrici di stringa di lunghezza prefissata: la lunghezza di ciascun elemento della matrice deve essere definita da un'istruzione DIM. Tuttavia possono sorgere dei problemi se gli elementi di una matrice di questo tipo sono lunghi, ad esempio, 20 caratteri. Infatti, se assegnamo ad uno di questi elementi una stringa di 15 caratteri, i 5 caratteri inutilizzati vengono riempiti con altrettanti spazi, che vanno eliminati quando la stringa viene usata per comporre una frase. La variabile A$, usata per trasferire alla subroutine un elemento della matrice, deve essere assegnato prima di chiamare la subroutine. Chiariamo il suo funzionamento con un esempio:

Versione Microsoft:

```
3650 SN$ = "IL TUO " + IV$(F,1) + " È INUTILE, LA FORZA
     CONTINUA A CRESCERE"
```

Versione Spectrum:

```
3650 LET S$ = "IL TUO ": A$ = IV$(F,1): GOSUB 8500: LET
     S$ = S$ + " È INUTILE, LA FORZA CONTINUA A CRE-
     SCERE"
```

Un secondo problema è la cancellazione dello schermo, che nella versione Commodore viene eseguita dall'istruzione PRINT CHR$(147). Sullo Spectrum è necessario sostituire questo comando con CLS.

Micro BBC:

Modificare il listato "Digitaya" inserendo le seguenti linee:

```
1400 A$ = GET$
1410 CLS
2630 REPEAT: A$ = GET$: UNTIL A$ = "S" OR A$ = "N"
2750 RA = RND(1)
2820 P = RND(40) + 7
3890 RD = RND(1): IF RD > .65 THEN 4110: REM COLPITO
4090 P = RND(40) + 7
4520 IV$(4,2) = STR$(RND(40) + 7): REM RIASSEGNA LA PO-
     SIZIONE DEL BIGLIETTO
4570 RN = RND(3) + 1
5560 RA = RND(1)
```

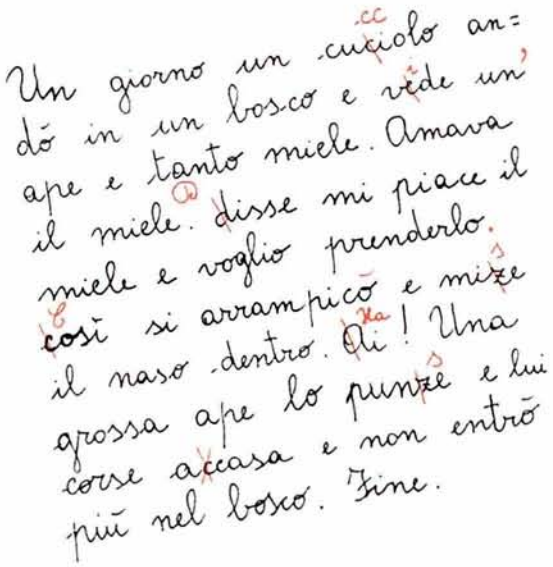
L'AVVENTURA DI DIGITAYA

```
1030 REM **       DIGITAYA       **
1040 REM ** UN GIOCO D'AVVENTURA **
1050 REM **     PER COMPUTER     **
1090 :
1100 GOSUB 6090: REM LEGGE I DATI DELLA MATRICE
1110 GOSUB 1290: REM PROLOGO DEL RACCONTO
1120 P=47: REM PUNTO INIZIALE
1130 :
1140 REM **** INIZIO DEL CICLO PRINCIPALE ****
1150 :
1160 MF=0: PRINT
1170 GOSUB 1440: REM DESCRIVE UN AMBIENTE
1180 GOSUB 1560: REM ELENCA LE USCITE POSSIBILI
1190 GOSUB 2670: REM P LOCAZIONE SPECIALE?
1200 IF SF=1 THEN 1250: REM CICLO SUCCESSIVO
1210 PRINT:INPUT "IL TUO COMANDO";IS$
1220 GOSUB 1700:REM ANALISI COMANDI
1225 IF F=0 THEN 1210:REM COMANDO NON VALIDO
1230 GOSUB 1900:REM COMANDI NORMALI
1240 IF VF=0 THEN PRINT "NON CAPISCO"
1250 IF MF=1 THEN 1160:REM NUOVA POSIZIONE
1260 IF MF=0 THEN 1210:REM NUOVO COMANDO
1270 END
1280 :
1290 REM **** PROLOGO DEL RACCONTO ****
1300 SN$="BENVENUTO A 'DIGITAYA'"
1310 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
1320 PRINT
1330 SN$="IL TUO APPARECCHIO 'BORBOTTA' SOTTOVOCE. TI GUARDI
     ATTORNO."
1340 SN$=SN$+" DA  NORD A SUD CORRE UNA STRADA DI GRANDE
     COMUNICAZIONE."
1350 SN$=SN$+" LA TUA MISSIONE E' TROVARE LA MISTERIOSA DIGITAYA"
1360 SN$=SN$+" E PORTARLA IN SALVO ATTRAVERSO UNA DELLE QUATTRO
     PORTE."
1370 SN$=SN$+" ... MA QUALE?"
1380 GOSUB 5880
1390 PRINT:PRINT "PREMI UN TASTO QUALSIASI PER GIOCARE"
1400 GET A$: IF A$=" " THEN 1400
1410 PRINTCHR$(147): REM CANCELLA LO SCHERMO
1420 RETURN
1430 :
1440 REM **** DESCRIVE UN AMBIENTE ****
1450 SN$="TI TROVI "+LN$(P): GOSUB 5880
1460 SN$="VEDI "
1470 REM ** RICERCA DELL OGGETTO **
1480 F=0:SP$=""
1490 FOR I=1TO8
1500 IF VAL(IV$(I,2))=P THEN SN$=SN$+SP$+"A "+IV$(I,1):F=1:SP$=" "
1510 NEXTI
1520 IF F=0 THEN SN$=SN$+"NESSUN OGGETTO"
1530 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
1540 RETURN
1550 :
1560 REM **** ELENCA LE USCITE POSSIBILI ****
1570 EX$=EX$(P)
1580 NR=VAL(LEFT$(EX$,2))
1590 ES=VAL(MID$(EX$,3,2))
1600 SD=VAL(MID$(EX$,5,2))
1610 OV=VAL(MID$(RIGHT$(EX$,2))
1620 IF (NR OR ES OR SD OR OV)=0 THEN RETURN
1630 PRINT:SN$="LE USCITE SONO A "
1640 IF NR <>0 THEN SN$=SN$+"NORD "
1650 IF ES <>0 THEN SN$=SN$+"EST "
1660 IF SD <>0 THEN SN$=SN$+"SUD "
1670 IF OV <>0 THEN SN$=SN$+"OVEST "
1675 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
1680 PRINT:RETURN
1690 :
1700 REM **** S/R ANALISI DEI COMANDI ****
1705 F=0:REM FLAG DI ZERO
1710 IF IS$="FINE" OR IS$="ELENCO" THEN VB$=IS$:F=1:RETURN
1720 IF IS$="GUARDA" THEN VB$=IS$:F=1:RETURN
1730 :
1740 REM ** SEPARA LE PAROLE **
1750 VB$=" ":NN$=" ":REM AZZERA VERBO E SOSTANTIVO
1770 LS=LEN(IS$)
1780 FOR C=1 TO LS
1790 A$=MID$(IS$,C,1)
1800 IF A$=" " THEN VB$=LEFT$(IS$,C-1):NN$=RIGHT$(IS$,LS-C)F=1:C=LS
1810 NEXT
1830 IF F=0 THEN PRINT"OCCORRONO ALMENO DUE PAROLE"
1840 RETURN
1850 :
1900 REM **** S/R AZIONI NORMALI ****
1910 VF=0
1920 PRINT
1930 IF VB$="VAI" OR VB$="MUOVI" THEN VF=1:GOSUB 2000
1940 IF VB$="PRENDI" OR VB$="RACCOGLI" THEN VF=1:GOSUB 2140
1950 IF VB$="LASCIA" OR VB$="DEPOSITA" THEN VF=1:GOSUB 2360
1960 IF VB$="ELENCA" OR VB$="INVENTARIO" THEN VF=1:GOSUB 2540
1965 IF VB$="GUARDA" THEN VF=1: MF=1: RETURN
1970 IF VB$="FINE" OR VB$="SMETTI" THEN VF=1:GOSUB 2610
1980 RETURN
1990 :
2000 REM **** S/R MOVIMENTO ****
2010 MF=1: REM SELEZIONA IL FLAG DI MOVIMENTO
2015 GOSUB 8600: REM RICERCA DELLA DIREZIONE
2020 DR$=LEFT$(NN$,1)
2030 IF DR$ <>"N" AND DR$ <>"E" AND DR$ <>"S" AND DR$<>"O"
     THEN 2100
2040 IF DR$="N" AND NR <>0 THEN P=NR: RETURN
2050 IF DR$="S" AND SD <>0 THEN P=SD: RETURN
2060 IF DR$="E" AND ES <>0 THEN P=ES: RETURN
2070 IF DR$="O" AND OV <>0 THEN P=OV: RETURN
2080 PRINT "NON PUOI ANDARE A ";IS$
2090 MF=0: RETURN
2100 REM COMANDO DI DIREZIONE ERRATO
2110 PRINT "CHE SIGNIFICA ";NN$;" ?"
2120 MF=0: RETURN
2130 :
2140 REM *** S/R PRENDI ***
2145 IV$(4,1)="BIGLIETTO PER IL TRI-STATO"
2150 GOSUB5730:REM OGGETTO VALIDO?
2160 IF F=0 THEN PRINT W$;" NON ESISTE":RETURN
2170 REM *** OGGETTO GIA' RACCOLTO? ***
2180 OV=F:GOSUB5830
2190 IFHF=1 THEN SN$="HAI GIA' RACCOLTO "+IV$(F,1):GOSUB5880:RETURN
2200 :
2210 REM *** L'OGGETTO E' QUI? ***
2220 IF VAL(IV$(F,2))<>P THENSN$=IV$(F,1)+" NON E' QUI":GOSUB5880:
     RETURN
2230 :
2240 REM ** AGGIUNGE OGGETTO ALL'ELENCO **
2250 AF=0:FOR J=1TO4
2260 IFIC$(J)=""THENIC$(J)=IV$(F,1):AF=1:J=4
2270 NEXT J
2280 :
2290 REM *** TROPPI OGGETTI? ***
2300 IF AF=0THENPRINT"HAI GIA' 4 OGGETTI":RETURN
2310 :
2320 SN$="HAI RACCOLTO "+IV$(F,1):GOSUB5880
2330 IV$(F,2)="-1":REM RIMUOVE OGGETTO DALL'ELENCO
2340 RETURN
2350 :
2360 REM ** S/R LASCIA **
2370 GOSUB 5730: REM OGGETTO VALIDO?
2380 IF F=0 THEN PRINT NN$;" NON ESISTE":RETURN
2390 :
2400 REM ** L'OGGETTO E' PRESENTE? **
2410 OV=F: GOSUB 5830
2420 IF HF=0 THEN PRINT "NON HAI CON TE ";IV$(F,1):RETURN
2430 :
2440 REM ** ABBANDONA L'OGGETTO **
2450 SN$="HAI ABBANDONATO "+IV$(F,1): GOSUB 5880
2460 IV$(F,2)=STR$(P): REM AGGIORNA POSIZ. OGGETTO
2470 :
2480 REM ** RIMUOVE OGGETTO DA QUELLI TRASPORTATI **
2490 FOR J=1 TO 4
2500 IF IC$(J)=IV$(F,1) THEN IC$(J)="": J=4
2510 NEXT J
2520 RETURN
2530 :
2540 REM *** S/R VISUALIZZA INVENTARIO ***
2550 PRINT "OGGETTI TRASPORTATI:"
2560 FOR I=1TO4
2570 PRINT"  ";IC$(I)
2580 NEXT I
2590 RETURN
2600 :
2610 REM ** S/R FINE GIOCO **
2620 PRINT:PRINT"SEI SICURO (S/N)?"
2630 GETA$:IFA$<>"S"AND A$<>"N"THEN2630
2640 IFA$="N"THEN RETURN
2650 END
2660 :
2670 REM **** S/R LOCAZIONE SPECIALE IN P ****
2680 SF=0: REM AZZERA FLAG PER LOCAZ. SPECIALI
2690 IF P=37 THEN 2780: REM TABELLA VETTORI
2700 IF P>7 THEN 2750: REM BUG CASUALE
2710 ON P GOSUB 2850, 2960, 3450, 3830, 4180, 4550, 5150
2720 RETURN
2730 :
2740 REM ** BUG CASUALE **
2750 RA=RND(TI)
2760 IF RA<0.05 THEN GOSUB 5420: REM CREA UN BUG
2770 RETURN
2780 REM ** TABELLA DEI VETTORI **
2790 SF=1
2800 SN$="TI SEI  SPOSTATO RAPIDAMENTE IN UN NUOVO AMBIENTE":
     GOSUB 5880
2810 FOR J=1 TO 1000: NEXT: REM PAUSA
2820 P=INT(RND(TI)*40+7)
2830 MF=1: RETURN
2840 :
2850 REM **** S/R USCITA DA TV ****
2860 SF=1
2870 SN$="HAI SCELTO L'USCITA TV E NON C'E' VIA DI SCAMPO."
2880 SN$=SN$+"SEI CONDANNATO A PARTECIPARE PER SEMPREAD UNO SHOW
     TELEVISIVO".
2890 GOSUB 5880: REM VISUALIZZA MESSAGGIO
2900 PRINT
2910 PRINT "BENVENUTO NELLO SHOW ....."
2920 FOR J=1 TO 500: NEXT J
2930 GOTO 2910
2940 END
2950 :
2960 REM **** S/R PORTA PER L'UTENTE ****
2970 SF=1
2980 SN$="LA FUGA E' A PORTATA DI MANO MA L'IMPIEGATO ALLA
     PRENOTAZIONE DDR"
2990 SN$=SN$+" TI SBARRA LA STRADA. TI INFORMA HA L'ORDINE DI
     FAR PASSARE"
3000 SN$=SN$+" SOLO INPUT. TUTTAVIA EGLI ACCETTA"
3010 SN$=SN$+" QUALSIASI CARTA DI CREDITO."
3020 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
3030 :
```

```
3040 PRINT: INPUT "COMANDI";IS$
3050 GOSUB 1700: REM ANALIZZA IL COMANDO
3060 GOSUB 1900: REM PROCEDURE NORMALI
3070 IF MF=1 THEN RETURN: REM ESCE
3080 IF VF=1 THEN 3040: REM ISTRUZIONE SUCCESSIVA
3090 IF VB$<>"CONSEGNA" THEN PRINT "NON CAPISCO": GOTO 3040
3100 :
3110 REM ** L'ISTRUZIONE E' 'CONSEGNA'**
3120 GOSUB 5730: REM OGGETTO VALIDO?
3130 IF F=0 THEN PRINT "NON HAI ";NN$: GOTO 3040:REM
     ISTRUZIONE SUCC.
3140 :
3150 REM ** L'OGGETTO E' UNA CARTA DI CREDITO? **
3160 IF F<>5 THEN PRINT "ACCETTA SOLO CARTE DI CREDITO":GOTO 3040
3170 :
3180 REM ** IL GIOCATORE POSSIEDE LA CARTA DI CREDITO? **
3190 OV=5: GOSUB 5830
3200 IF HF=0 THEN PRINT "NON HAI ";IV$(5,1): GOTO 3040
3210 :
3220 SN$="L'IMPIEGATO PRENDE LA CARTA E DICE 'MOLTO BENE, SIGNORE'"
3230 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
3240 SN$="OLTREPASSI LA BARRIERA ED ENTRI NELLA PORTA PER L'UTENTE"
3250 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
3260 :
3270 REM ** IL GIOCATORE POSSIEDE DIGITAYA? **
3280 OV=6: GOSUB 5830
3290 IF HF=1 THEN 3380: REM MISSIONE COMPIUTA
3300 :
3310 REM ** INSUCCESSO **
3320 SN$="SEI RIUSCITO A SFUGGIRE ALLA MACCHINA"
3330 SN$=SN$+" MA LA TUA MISSIONE E' FALLITA PERCHE' NON HAI"
3340 SN$=SN$+" PORTATO CON TE IL MISTERIOSO DIGITAYA"
3350 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
3360 END
3370 :
3380 REM ** MISSIONE COMPIUTA **
3390 SN$="CONGRATULAZIONI, SEI RIUSCITO A LIBERARE"
3400 SN$=SN$+" IL MERAVIGLIOSO DIGITAYA DALLA"
3410 SN$=SN$+" MORSA DELLA MACCHINA."
3420 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
3430 END
3440 :
3450 REM ** S/R PORTA PER CASSETTA **
3460 SF=1
3470 SN$="SENTI UNA FORZA IRRESISTIBILE CHE TI TRASCINA VERSO "
3480 SN$=SN$+"UNO STATO DI AMNESIA MAGNETICA PERMANENTE"
3490 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
3500 NS=0: REM ISTRUZIONI DI INIZIO CONTEGGIO
3510 REM ** ISTRUZIONI **
3520 NS=NS+1: IF NS>3 THEN 3770: REM ESPULSO
3530 PRINT: INPUT "COMANDI";IS$
3540 GOSUB 1700: REM ANALIZZA IL COMANDO
3550 GOSUB 1900: REM PROCEDURE NORMALI
3560 IF MF=1 THEN MF=0: PRINT "NON RIESCI A MUOVERTI":GOTO 3510
3570 IF VF=1 THEN 3510: REM ISTRUZIONE SUCC.
3580 IF VB$<>"USA" THEN PRINT "NON CAPISCO": GOTO 3510
3590 REM ** L'ISTRUZIONE E' 'USA' **
3600 GOSUB 5730: REM OGGETTO VALIDO?
3610 IF F=0 THEN PRINT "NON HAI ";NN$: GOTO 3510
3620 :
3630 REM ** L'OGGETTO E' UN ATTIVATORE DEL BUFFER? **
3640 IF F=8 THEN 3680: REM TUTTO OK
3650 SN$="IL TUO "+IV$(F,1)+"E' INUTILE, L'ATTRAZIONE AUMENTA"
3660 GOSUB 5880: GOTO 3510 :REM ISTRUZIONE SUCCESSIVA
3670 :
3680 OV=8: GOSUB 5830: REM IL GIOCATORE POSSIEDE L'ATTIVATORE?
3690 IF HF=0 THEN SN$="NON POSSIEDI "+IV$(8,1):GOSUB 5880:
     GOTO 3510
3700 :
3710 REM ** IN SALVO **
3720 SN$="USI L'ATTIVATORE DEL BUFFER PER OPPORTI"
3730 SN$=SN$+" ALL'AMNESIA MAGNETICA. LA FORZA DIMINUISCE"
3740 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
3750 RETURN
3760 :
3770 REM ** ESPULSO **
3780 SN$="LA FORZA DIVIENE INSOSTENIBILE E VIENI ESPULSO
     NEL 'NULLA'"
3790 SN$=SN$+" MAGNETICO ATTRAVERSO LA PORTA PER LA CASSETTA."
3800 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
3810 END
3820 :
3830 REM **** LOCAZIONE: PORTA DEL JOYSTICK ****
3840 SF=1
3850 SN$="UN UTENTE DAGLI OCCHI ARROSSATI TI CENTRA  PIU' VOLTE
     COL SUO LASER."
3860 GOSUB 5880: REM VISUALIZZA MESSAGGIO
3870 :
3880 REM ** IMMISSIONE COMANDI **
3890 RD=RND(TI): IF RD>.65 THEN 4110: REM COLPITO!
3900 PRINT:INPUT "IL TUO COMANDO";IS$
3910 GOSUB 1700: GOSUB 1900: REM ESAMINA IL COMANDO
3920 IF MF=1 THEN MF=0: PRINT "NON PUOI MUOVERTI...PER ORA":
     GOTO 3880
3930 IF VF=1 THEN 3880: REM COMANDO SUCCESSIVO
3940 IF VB$<>"USA" THEN PRINT "NON CAPISCO": GOTO 3880
3950 GOSUB 5730: REM OGGETTO VALIDO
3960 IF F=0 THEN PRINT NN$;" NON ESISTE ": GOTO 3880: REM
     COMANDO SUCCESSIVO
3970 :
3980 REM ** L'OGGETTO E' UNO SCUDO ANTI-LASER **
3990 IF F=3 THEN 4020: REM TUTTO OK
4000 SN$="IL TUO "+IV$(F,1)+" NON FUNZIONA": GOSUB 5880: GOTO 3880
4010 :
4020 OV=3: GOSUB 5830: REM OGGETTO E' LO SCUDO ANTI-LASER
4030 IF HF=0 THEN SN$="NON POSSIEDI LO "+IV$(3,1): GOSUB 5880:
     GOTO 3880
4040 :
4050 REM ** SALVO! **
4060 SN$="LO SCUDO TI DIFENDE. COLPI A RAFFICA."
4070 SN$=SN$+"ESCI DALLA PORTA DEL JOYSTICK E RIENTRI NELLA
     MACCHINA."
4080 GOSUB 5880: REM VISUALIZZA MESSAGGIO
4090 P=INT(RND(TI)*40+7): MF=1: RETURN
4100 :
4110 REM ** COLPITO! **
4120 SN$="SEI  STATO DISINTEGRATO DAL LASER E TE  NE  RENDI
     CONFUSAMENTE CONTO"
4130 SN$=SN$+" GLI ATOMI DEL TUO CORPO SI SONO SPARSI"
4140 SN$=SN$+" PER TUTTO L'UNIVERSO"
4150 GOSUB 5880: REM VISUALIZZA MESSAGGIO
4160 END
4170 :
4180 REM **** S/R DISPOSITIVO TRI-STATO ****
4190 SF=1
4200 SN$="UN GRANDE CARTELLO ANNUNCIA 'STAZIONE DI INGRSSO/USCITA'
     MA, APPENA TI AVVICINI,"
4210 SN$=SN$+" UN CONTROLLORE GRIDA 'BIGLIETTO, PREGO'"
4220 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
4230 :
4240 REM ** ISTRUZIONI **
4250 PRINT: INPUT "COMANDI";IS$
4260 GOSUB 1700: GOSUB 1900: REM ANALIZZA
4270 IF MF=1 THEN RETURN
4280 IF VF=1 THEN 4240: REM ISTRUZIONE SUCCESSIVA
4290 IF VB$<>"MOSTRA" AND VB$<>"CONSEGNA" THEN PRINT "NON CAPISCO":
     GOTO 4240
4300 REM ** L'ISTRUZIONE E' 'MOSTRA' **
4310 GOSUB 5730: REM OGGETTO VALIDO?
4320 IF F=0 THEN PRINT "NON HAI ";NN$: GOTO 4240: REM ISTRUZIONE
     SUCCESSIVA
4330  :
4340 REM ** L'OGGETTO E' UN BIGLIETTO? **
4350 IF F=4 THEN 4400: REM TUTTO OK
4360 SN$="IL CONTROLLORE SCUOTE LA TESTA E DICE"
4370 SN$=SN$+" 'QUESTO "+IV$(F,1)+" NON E' VALIDO'"
4380 GOSUB 5880: GOTO 4240: REM ISTRUZIONE SUCCESSIVA
4390 :
4400 OV=4: GOSUB 5830: REM IL GIOCATORE POSSIEDE IL BIGLIETTO?
4410 IF HF=0 THEN PRINT "NON HAI IL BIGLIETTO": GOTO 4240
4420 :
4430 REM ** TUTTO OK **
4440 SN$="IL CONTROLLORE FORA IL BIGLIETTO E TI PERMETTE"
4450 SN$=SN$+" DI SUPERARE LA BARRIERA."
4460 GOSUB 5880: REM VISUALIZZA IL MESSAGGIO
4470 REM ** CANCELLA IL BIGLIETTO DALL'INVENTARIO **
4480 F=0
4490 FOR J=1 TO 4
4500 IF IC$(J)=IV$(4,1) THEN IC$(J)="": J=4
4510 NEXT J
4520 IV$(4,2)=STR$(INT(RND(TI)*40+8)): REM RIDEFINISCE LA POSIZIONE
     DEL BIGLIETTO
4530 P=15: MF=1: RETURN
4540 :
4550 REM **** UNITA' LOGICO-ARITMETICA (ALU) ****
4560 SF=1
4570 RN=INT(RND(TI)*3+1)
4580 IF RN=1 THEN CD$="AND"
4590 IF RN=2 THEN CD$="OR"
4600 IF RN=3 THEN CD$="NOT"
4610 SN$="SULLA PARETE CI SONO TRE PULSANTI CONTRASSEGNATI"
4620 SN$=SN$+" DALLE  SCRITTE 'AND', 'OR' E  'NOT'. PREMENDO IL
     PULSANTE GIUSTO"
4630 SN$=SN$+" PUOI ENTRARE NELL'ACCUMULATORE"
4640 GOSUB 5880: REM VISUALIZZA MESSAGGIO
4650 :
4660 REM ** IMMISSIONE COMANDI **
4670 PRINT:INPUT "IL TUO COMANDO";IS$
4680 GOSUB 1700: GOSUB 1900: REM ANALISI COMANDO
4690 IF MF=1 THEN RETURN: REM USCITA
4700 IF VF=1 THEN 4670: REM COMANDO SUCCESSIVO
4710 IF VB$="USA" OR VB$="PREMI" THEN 4740
4720 PRINT "NON CAPISCO": GOTO 4670
4730 :
4740 REM ** COMANDO VALIDO **
4750 IF VB$="PREMI" THEN 4930
4760 REM ** COMANDO E' 'USA'? **
4770 GOSUB 5730: REM OGGETTO VALIDO?
4780 IF F=0 THEN PRINT NN$;" NON ESISTE":GOTO 4670: REM COMANDO
     SUCCESSIVO
4790 :
4800 REM ** L'OGGETTO E' UN MANUALE DEL CODICE? **
4810 IF F=7 THEN 4850: REM TUTTO OK
4820 SN$="QUI "+IV$(F,1)+" E' DEL TUTTO INUTILE": GOSUB 5880
4830 GOTO 4670: REM COMANDO SUCCESSIVO
4840 :
4850 OV=7: GOSUB 5830: REM IL MANUALE E' TRASPORTATO?
4860 IF HF=1 THEN 4900: REM TUTTO OK
4870 SN$="NON POSSIEDI "+IV$(7,1)
4880 GOSUB 5880: GOTO 4670: REM COMANDO SUCCESSIVO
4890 :
4900 SN$="APRI IL MANUALE E TROVA LA PAROLA '"+CD$+"'"
4910 GOSUB 5880: GOTO 4670: REM COMANDO SUCCESSIVO
4920 :
4930 REM ** IL COMANDO E' 'PREMI' **
4940 IF NN$="AND" OR NN$="OR" OR NN$="NOT" THEN 4970
4950 SN$=NN$+" NON ESISTE": GOSUB 5880: GOTO 4670: REM COMANDO
     SUCCESSIVO
4960 :
4970 REM ** PULSANTE GIUSTO O SBAGLIATO? **
```

IAN McKINNELL

```
4980 IF NN$=CD$ THEN GOSUB 5100: RETURN
4990 GOSUB 5010: RETURN
5000 :
5010 REM ** S/R PULSANTE SBAGLIATO **
5020 SN$="SBAGLIATO! SI APRE  UNA BOTOLA E  PRECIPITI"
5030 SN$=SN$+" TORNANDO NELLA MEMORIA PRINCIPALE"
5040 GOSUB 5880: REM VISUALIZZA MESSAGGIO
5050 IF RN=1 THEN P=39
5060 IF RN=2 THEN P=35
5070 IF RN=3 THEN P=29
5080 MF=1: RETURN
5090 :
5100 REM ** S/R PULSANTE GIUSTO **
5110 SN$="SI APRE LA PORTA DELL'ACCUMULATORE"
5120 SN$=SN$+" E TU L'ATTRAVERSI": GOSUB 5880
5130 P=30: MF=1: RETURN
5140 :
5150 REM **** S/R INGRESSO IN MEMORIA ****
5160 SF=1
5170 SN$="UN GUARDIANO TI INFORMA CHE NON PUOI ENTRARE "
5180 SN$=SN$+"SE NON DAI UN INDIRIZZO": GOSUB 5880
5190 REM ** IMMISSIONE COMANDO **
5200 PRINT:INPUT "IL TUO COMANDO";IS$
5210 GOSUB 1700: GOSUB 1900: ESAMINA IL COMANDO
5220 IF MF=1 THEN RETURN: REM USCITA
5230 IF VF=1 THEN 5200: REM COMANDO SUCCESSIVO
5240 IF VB$<>"DAI" THEN PRINT "NON CAPISCO": GOTO 5200
5250 :
5260 GOSUB 5730: REM OGGETTO VALIDO?
5270 IF F=0 THEN PRINT NN$;" NON ESISTE": GOTO 5200: REM COMANDO
     SUCCESSIVO
5280 :
5290 REM ** L'OGGETTO E' UN INDIRIZZO? **
5300 IF F=1 THEN 5330: REM TUTTO OK
5310 PRINT "HA BISOGNO DEL TUO INDIRIZZO": GOTO 5200
5320 :
5330 OV=1: GOSUB 5830: REM INDIRIZZO TRASPORATO?
5340 IF HF=1 THEN 5370
5350 SN$=IV$(1,1)+ " NON E' IN TUO POSSESSO":GOSUB 5880: GOTO 5200
5360 :
5370 REM ** INGRESSO IN MEMORIA CONSENTITO **
5380 SN$="IL GUARDIANO ESAMINA IL TUO INDIRIZZO E TI LASCIA ENTRARE"
5390 SN$=SN$+" ATTRAVERSO": GOSUB 5880
5400 P=40: MF=1: RETURN
5410 :
5420 REM ** BUG CASUALE **
5430 SF=1
5440 SN$="DA DIETRO UN CHIP APPARE UN GRANDE E MINACCIOSO BUG"
5450 SN$=SN$+" CHE TI AGGREDISCE": GOSUB 5880
5460 :
5470 REM ** IMMISSIONE COMANDO **
5480 PRINT:INPUT "IL TUO COMANDO";IS$
5490 GOSUB 1700: GOSUB 1900: REM ANALISI COMANDO
5500 IF MF=1 THEN MF=0: PRINT "NON RIESCI A MUOVERTI ...": GOTO 5480
5510 IF VF=1 THEN 5480: REM COMANDO SUCCESSIVO
5520 IF VB$="UCCIDI" OR VB$="LOTTA" THEN 5550
5530 PRINT "NON CAPISCO": GOTO 5480
5540 :
5550 REM ** IL COMANDO E' UCCIDERE O LOTTARE **
5560 RA=RND(TI)
5570 IF RA<0.5 THEN GOSUB 5600
5580 GOSUB 5670: RETURN
5590 :
5600 REM **** S/R VIENI UCCISO ****
5610 SN$="INIZIA LA LOTTA. IL BUG TI LANCIA CONTRO ERRORI
     'PARASSITARI'"
5620 SN$=SN$+" CHE DISTRUGGONO IL TUO CERVELLO."
5630 SN$=SN$+" IL DOLORE  E' INSOPPORTABILE ED INFINE LA TUA
     TESTA ESPLODE."
5640 GOSUB 5880
5650 END
5660 :
5670 REM **** S/R UCCIDI IL BUG ****
5680 SN$="TI DIFENDI  DISPERATAMENTE ...."
5690 SN$=SN$+" INFINE RIESCI AD ABBATTERE IL BUG  E SOPRAVVIVI.":
     GOSUB 5880
5700 RETURN
5710 :
5720 :
5730 REM *** S/R OGGETTO VALIDO ***
5740 NN$=NN$+" ":LN=LEN(NN$):F=0:C=1
5745 FOR K=1 TO LN
5750 IF MID$(NN$,K,1)<>" " THEN NEXTK:RETURN
5755 W$=MID$(NN$,C,K-C):C=K+1:LW=LEN(W$)
5760 FORJ=1 TO 8
5770 LI=LEN(IV$(J,1)):REM LUNGHEZZA OGGETTO
5780 FOR I=1TO LI
5790 IF MID$(IV$(J,1),I,LW)=W$ THEN F=J:I=LI:J=8:K=LN
5800 NEXT I,J,K
5810 RETURN
5820 :
5830 REM *** S/R OGGETTO TRASPORTATO? ***
5840 HF=0
5850 IF IV$(OV,2)="-1"THEN HF=1
5860 RETURN
5870 :
5880 REM **** S/R FORMATTAZIONE STAMPA ****
5890 CL=0: REM CONTATORE DI LINEA/CARATTERE
5900 VC=1: REM CONTEGGIO PRECEDENTE
5910 PV$=" ": REM PAROLA PRECEDENTE
5920 LL=40: REM LUNGHEZZA LINEA SCHERMO
5930 SN$=SN$+" PIPPO " :REM VARIABILE DI COMODO
5940 PRINT
5950 FOR C=1 TO LEN(SN$)
5960 CL=CL+1
5970 IF MID$(SN$,C,1) = " " THEN GOSUB 6020
5980 NEXT C
5990 PRINT
6000 RETURN
6010 :
6020 REM **** S/R VERIFICA DI FINE LINEA ****
6030 PN$=MID$(SN$,VC,C-VC+1): REM PAROLA NUOVA
6040 IF CL<LL THEN PRINT PV$; : GOTO 6060
6050 PRINT PV$:CL=LEN(PN$)
6060 VC = C + 1:PV$ = PN$
6070 RETURN
6080 :
6090 REM ** S/R LETTURA MATRICE DEI DATI **
6100 REM * LETTURA INVENTARIO *
6110 DIM IV$(8,2),IC$(4)
6120 FOR C=1 TO 8
6130 READ IV$(C,1),IV$(C,2)
6140 NEXT C
6150 :
6160 REM * LETTURA AMBIENTE E USCITE *
6170 DIM LN$(55),EX$(55)
6180 C1=0:C2=0:REM INIZIALIZZA LE  CHECKSUM
6190 FOR C=1 TO 54
6200 READ LN$(C),EX$(C)
6210 C1=C1+VAL(LEFT$(EX$(C),4))
6220 C2=C2+VAL(RIGHT$(EX$(C),4))
6230 NEXT C
6240 READ CA:IFCA<>C1THEN PRINT"ERRORE CHECKSUM":STOP
6250 READ CB:IFCB<>C2THEN PRINT"ERRORE CHECKSUM":STOP
6260 RETURN
6270 REM ** DATI PER L'INVENTARIO **
6280 DATA NUMERO INDIRIZZO,45,CHIAVE,34,SCHERMO LASER,25
6290 DATA BIGLIETTO PER TRI-STATO,26,CARTA DI CREDITO PER DATI,28
6300 DATA DIGITYA,30,CIFRARIO,19,DISPOSITIVO DI AZIONAMENTO
     BUFFER,13
6310 :
6320 REM ** DATI SU AMBIENTI E USCITE **
6330 DATA NEL CAVO TV,00000000
6340 DATA NELLA PORTA UTENTE,00090100
6350 DATA NELLA PORTA DEL REGISTRATORE,00110000
6360 DATA NELLA PORTA DEL JOYSTICK,00130000
6370 DATA IN UN DISPOSITIVO TRI-STATO,00170000
6380 DATA "NELL'ULA",00310016
6390 DATA "ALL'INGRESSO DELLA MEMORIA",00490000
6400 DATA "SULL'AUTOSTRADA DI I/O",09000001
6410 DATA "SULL'AUTOSTRADA DI I/O",10000802
6420 DATA "SULL'AUTOSTRADA DI I/O",11000900
6430 DATA "SULL'AUTOSTRADA DI I/O",12001003
6440 DATA "SULL'AUTOSTRADA DI I/O",13531100
6450 DATA "SULL'AUTOSTRADA DI I/O",14001204
6460 DATA "SULL'AUTOSTRADA DI I/O",15001300
6470 DATA "SULL'AUTOSTRADA DI I/O UN SEGNALE DICE 'S OUT H'",
     00001400
6480 DATA NEL REGISTRO DEI DATI,00061700
6490 DATA "SU UNA AUTOSTRADA A 8 CORSIE",16001805
6500 DATA "SU UNA AUTOSTRADA A 8 CORSIE",17001900
6510 DATA "SU UNA AUTOSTRADA A 8 CORSIE",18002000
6520 DATA "SU UNA AUTOSTRADA A 8 CORSIE",19292100
6530 DATA "SU UNA AUTOSTRADA A 8 CORSIE",20282200
6540 DATA "SU UNA AUTOSTRADA A 8 CORSIE",21272300
6550 DATA "SU UNA AUTOSTRADA A 8 CORSIE",22262400
6560 DATA "SU UNA AUTOSTRADA A 8 CORSIE",23250000
6570 DATA NELLA MATRICE DEI CARATTERI,26360024
6580 DATA IN ALTO NELLA MEMORIA,27352523
6590 DATA IN MEZZO DELLA MEMORIA,28342622
6600 DATA IN MEZZO DELLA MEMORIA,29332721
6610 DATA IN BASSO NELLA MEMORIA,00542820
6620 DATA NELLA TANA DELL'ACCUMULATORE,00000600
6630 DATA IN UN LUNGO CORRIDOIO,00420006
6640 DATA IN UN REGISTRO INDICE,31000000
6650 DATA BASSO NELLA MEMORIA,54403428
6660 DATA NEL MEZZO DELLA MEMORIA,33393527
6670 DATA IN ALTO NELLA MEMORIA,34383626
6680 DATA NELLA MATRICE DEI CARATTERI,35370025
6690 DATA IN UNA TABELLA DI VETTORI CASUALI,00000000
6700 DATA "IN ALTO NELLA MEMORIA, SOPRA UN'AUTOSTRADA",39003735
6710 DATA NEL MEZZO DELLA MEMORIA,40003834
6720 DATA NELLA MEMORIA - AD EST C'E' UN CANCELLO,41003933
6730 DATA IN BASSO NELLA MEMORIA,00004054
6740 DATA IN UN CORRIDOIO,00430031
6750 DATA IN UN CORRIDOIO,00440042
6760 DATA IN UN CORRIDOIO,00004543
6770 DATA NEL REGISTRO DEGLI INDIRIZZI,00004600
6780 DATA "SU UN'AUTOSTRADA A 16 CORSIE,45004700
6790 DATA "SU UN'AUTOSTRADA A 16 CORSIE,46004800
6800 DATA "SU UN'AUTOSTRADA A 16 CORSIE,47004900
6810 DATA "SU UN'AUTOSTRADA A 16 CORSIE C'E' UN GRANDE CANCELLO
     A OVEST,48005007
6820 DATA "SU UN'AUTOSTRADA A 16 CORSIE,49005100
6830 DATA "SU UN'AUTOSTRADA A 16 CORSIE,50005200
6840 DATA "SU UN'AUTOSTRADA A 16 CORSIE,51000000
6850 DATA IN UN VETTORE DI MEMORIA,00290012
6860 DATA IN BASSO NELLA MEMORIA,00413329
6870 REM ** DATI DI CHECKSUM **
6880 DATA 100169,103973
8599 :
8600 REM ** S/R RICERCA DELLA DIREZIONE **
8610 NN$=NN$+" ":LN=LEN(NN$):C=1
8620 FOR I=1 TO LN
8630 IF MID$(NN$,I,1)<>" " THEN NEXT I:RETURN
8640 W$=MID$(NN$,C,I-C):C=I+1:
8650 IF W$="NORD" OR W$="EST" THEN NN$=W$:I=LN
8660 IF W$="SUD" OR W$="OVEST" THEN NN$=W$:I=LN
8670 NEXT I
8680 RETURN
```


Un giorno un cucciolo andò in un bosco e vede un'ape e tanto miele. Amava il miele. disse mi piace il miele e voglio prenderlo. così si arrampicò e mise il naso dentro. Ai! Una grossa ape lo punze e lui corse accasa e non entrò più nel bosco. Fine.

Un giorno un cucciolo ando' in un bosco e vede un'ape e tanto miele. Amava il miele. disse mi piace il miele e voglio prenderlo cosi si arrampico e mize il naso dentro. Ai! Una grossa ape lo punze e lui corse accasa e non entro' piu' nel bosco. Fine.

Scrivere col computer

Può un micro aiutare i più piccoli nell'apprendimento della scrittura?

Le innovazioni tecniche, come del resto tutte le innovazioni in genere, incontrano forti difficoltà iniziali prima di essere accettate. Il quaderno, ad esempio, è oggi un oggetto indispensabile per qualsiasi studente, tuttavia rappresentò un'innovazione di vaste proporzioni quando ne venne introdotto l'uso al posto della vecchia lavagna. Prima, infatti, la carta veniva adoperata soprattutto per registrare transazioni commerciali o per redigere importanti documenti civili o religiosi: farla usare ai bambini sembrava, forse, un terribile spreco.
Al contrario, il quaderno offre la possibilità di avere una traccia permanente, da consultare e riguardare, sui propri progressi in una materia di studio. Benché si sia trattato inizialmente di un'innovazione costosa, si è poi dimostrata di incalcolabile valore pratico. Oggi la storia si ripete nei confronti dei microcomputer: molti ritengono totalmente assurdo mettere un computer a disposizione di ciascun studente. Indubbiamente, sotto il profilo economico, che purtroppo predomina sempre in questo particolare settore, l'ostacolo è insormontabile.
Inoltre c'è spesso da superare la resistenza da parte di alcuni insegnanti un poco restii ad introdurre uno strumento così innovativo, specialmente dopo la cattiva fama creatasi attorno alla "istruzione programmata", di cui si è parlato in precedenza.
Un'attenta analisi sulle effettive applicazioni del computer in classe, dunque, può servire a fugare molti dei più diffusi pregiudizi e a dimostrare che, a lungo termine, il computer non solo accresce l'efficienza e l'efficacia dell'insegnamento (dimostrandosi così un investimento molto più produttivo di quanto possa sembrare inizialmente), ma riesce anche a stimolare una maggiore creatività espressiva negli allievi.

Creatività letteraria

Consideriamo per primo l'impiego di computer come aiuto nello sviluppo di capacità espressive tramite scrittura. Concepire la trama di un racconto è, per molti versi, il passo più facile: il grosso del lavoro consiste nell'esprimere il tutto in una forma scritta definiti-

Giocare con le parole

Uno dei maggiori problemi che i bambini devono affrontare, nel tentativo di esprimersi creativamente, è dovuto al tempo necessario per l'azione di scrivere, che spesso non riesce a star dietro al corso dei pensieri. Dovendo applicarsi nel tracciare la corretta forma di ciascuna lettera si perde facilmente il tenue filo di un'idea. Spesso lo spezzettamento logico di un componimento deriva proprio da questo motivo. Il semplice uso di un word processor non può rimuovere questo inconveniente: il tempo e l'attenzione necessari per trovare una lettera sulla tastiera porta a risultati pressoché identici. Tuttavia, se al bambino viene insegnata la dattilografia, il processo diviene più meccanico ed interferisce meno sul risultato finale

Sviluppare le proprie doti

L'uso di un computer per redigere testi (tramite word processor) e per tracciare disegni sviluppa molte doti, alcune delle quali sono comuni anche ai metodi di scrittura e disegno tradizionali, ma altre completamente nuove. Qui a fianco vengono riassunte quelle che vengono maggiormente stimolate

WORD PROCESSOR

- Ottima capacità motoria (mobilità delle dita, in particolare del mignolo)
- Riconoscimento delle lettere
- Formazione delle lettere (la riproduzione dei singoli caratteri, su video e stampante, è omogenea e può insegnare a riprodurla anche a mano)
- Correzione delle "bozze" e verifica degli errori
- Compitazione
- Collocazione di frasi nel loro ordine logico
- Memoria visiva e meccanica (imparare a trovare i tasti senza guardare la tastiera)

Programmi di grafica

- Coordinazione tra mani e occhi
- Ottima capacità motoria
- Creazione di motivi ricorrenti
- Discernimento delle forme
- Relazioni di causa/effetto
- Scorrevolezza (le idee vengono fissate rapidamente)
- Elasticità mentale (analisi di forme familiari sotto nuovi aspetti)
- Immaginazione e creatività



va e ciò richiede molti aggiustamenti, riscritture e correzioni, e talvolta si deve ricominciare tutto da capo. Il processo è comune a qualsiasi prodotto "artistico", sia esso un dipinto o una statua.

Sfortunatamente, nella scuola attuale, il metodo di scrittura col quale ci si aspetta che i bambini esprimano le proprie idee finisce con l'ostacolare i processi creativi. Nella maggior parte dei casi la prima imposizione da parte dell'insegnante è quella di dover consegnare il compito nel minor tempo possibile e, per giunta, scritto in bella copia, senza cancellature, rimandi e via dicendo. L'aspetto estetico (di forma e non di contenuto), che pure ha una sua importanza, rischia troppo spesso di prevalere su quello espressivo e creativo.

L'uso di un word processor può indubbiamente curare questa sindrome della "bella copia", facendo risparmiare tempo a tutto vantaggio dello studente, il quale impara a riguardare e a correggere il proprio lavoro, operando ogni sorta di interventi, pur producendo una versione finale "pulita". Il testo, conservato in un file su disco, viene consegnato all'insegnante che può poi rivederlo e correggerlo su un altro computer, magari anche a casa propria. Se invece i computer sono collegati in rete, l'insegnante può in qualsiasi momento richiamare il file dell'allievo, anche per controllarne via via lo sviluppo. Questo senza contare che talvolta esistono programmi in grado di controllare e correggere automaticamente l'ortografia di un componimento archiviato in un file.

Ovviamente, anche questo metodo automatizzato non è del tutto perfetto: evidenziare su carta gli errori commessi è probabilmente più incisivo ed efficace. La correzione, sempre su carta, è più facile e comoda, e inoltre consente note e commenti "a margine".

Quando viene proposto l'uso di computer nell'insegnamento, c'è sempre chi solleva il problema della barriera psicologica dovuta alla tastiera dell'apparecchio che, a detta di molti, scoraggia chi si avvicina per la prima volta a un computer. Raramente però ci si rende conto che esiste anche una barriera della matita: il processo di apprendimento della scrittura tradizionale infatti è assai laborioso e difficoltoso. Per prima cosa occorre imparare a tenere la matita nella posizione corretta, poi imparare bene le forme delle lettere e a riprodurle correttamente su carta. Le lettere vanno disegnate accuratamente e seguendo un certo ordine e, inoltre, il bambino deve imparare a scrivere le parole in grandezza uniforme e lungo una linea diritta (non solo in maiuscolo, ma anche in corsivo). Tutto ciò richiede anni di esercizio, benché la calligrafia rimanga spesso di qualità discutibile.

Si confronti questo complesso procedimento con quello necessario per imparare la dattilografia: basta saper riconoscere la forma delle lettere e imparare dove queste sono collocate sulla tastiera adoperando, per la loro pressione, una precisa impostazione delle dita.

Senza dubbio questo metodo è molto più facile.



Differenze artistiche

Questi disegni sono stati fatti da Timothy Ginn di otto anni. Tutti dimostrano alcune caratteristiche comuni: accuratezza nei particolari, una separazione tra le varie forme, assieme a una corretta collocazione relativa degli oggetti, in proporzione tra loro. Tuttavia il terzo disegno, creato con il MacPaint su di un Macintosh, è un miglioramento qualitativo: è dotato di una certa prospettiva, di uno sfondo (inesistente negli altri due disegni) e di movimento (le particelle nella scia dell'astronave). Simili differenze possono essere ancora più evidenti, specie nei bambini che hanno difficoltà con i mezzi espressivi tradizionali; essi riescono meglio a tradurre in immagini ciò che esiste nella loro immaginazione

Con ciò nessuno si immagina di voler abolire la scrittura a tradizionale.
Avendo un limite di tempo prefissato, anche l'allievo con maggiori doti espressive è condizionato dalle proprie capacità ortografiche, ma insegnandogli la dattilografia (magari in tenera età) i risultati possono sorprendere notevolmente, sia per qualità che per quantità.
La scrittura tradizionale ha pur sempre un ruolo ben preciso, ma non va trascurato il fatto che l'uso della tastiera, in particolare nel mondo del lavoro, è ormai quasi una regola che ha solide radici: prima s'impara la dattilografia, dunque, meglio è.

Il disegno

Un altro aspetto della diffusione dei microcomputer nella scuola è la facilità con la quale si possono creare, sullo schermo del computer, sofisticate immagini grafiche. Inizialmente il costo degli apparecchi che consentivano di sfruttare appieno questa applicazione era proibitivo, ma oggi si ottengono ottimi risultati anche con computer dal costo molto accessibile.
Da una parte esistono varie tavolette grafiche (vedere alle pagg. 1109 e 1310) e dall'altra i vari mouse. Entrambi questi dispositivi, da applicare al computer, favoriscono la produzione di immagini colorate sullo schermo. Nel caso dei mouse si va ancora oltre, poiché in certi casi le funzioni dello stesso computer possono essere selezionate con esso (è il caso del Macintosh della Apple).
Per di più, lo studente viene messo in grado di combinare, nelle proprie realizzazioni, testi e grafica con criteri molto più semplici di quelli tradizionali. Ad esempio, una particolare figura preparata in precedenza può essere facilmente richiamata e inserita a piacimento in un punto particolare di un testo redatto tramite word processor.
Ciò nonostante i programmi per computer, per quanto sofisticati siano, non potranno mai sostituire completamente gli strumenti artistici tradizionali, che offrono comunque agli allievi un nuovo mezzo espressivo col quale sperimentare ed imparare nuove forme per esprimere le proprie idee e intuizioni.
I bambini vogliono imparare: la loro innata curiosità li spinge a esplorare, a ricercare e ad apprendere quanto più possibile sull'ambiente che li circonda. Purché non si limiti questo impulso spontaneo, i bambini riescono poi a elaborare mentalmente le proprie esperienze e ad esprimerle in modo creativo (spesso, ad esempio, afferrando un qualsiasi oggetto che scrive e magari producendo improvvisati "murales").
Perfino un bambino di due anni, posto di fronte a un computer, tenta di esplorarlo premendo i tasti e seguendo quanto accade sullo schermo: egli accetta con estrema naturalezza tutte quelle che per gli adulti rappresentano innovazioni tecniche, ma che per lui rappresentano soltanto l'ambiente che lo circonda.

Tipi di variabili

Il PASCAL prevede quattro tipi diversi di variabili: vediamone le caratteristiche

Sagome simboliche

Queste sono le tre figure generalmente impiegate nei diagrammi sintattici del PASCAL:



- Questa forma ovoidale rappresenta parole riservate del PASCAL oppure caratteri che non richiedono ulteriore spiegazione (come "lettera" o "numero")

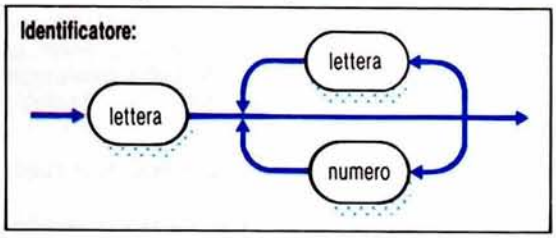


- Un cerchio identifica un operatore del PASCAL (ad esempio: +, -, *, . , ecc.)



- Il rettangolo rappresenta una parola o una frase che possiede un diagramma sintattico definito a parte

Ai quattro tipi di variabili predefiniti, in PASCAL, sono attribuiti i nomi: Integer, Real, Char e Boolean. I primi due si riferiscono a valori numerici, distinguendo quelli che possiedono una parte decimale (numeri "reali") da quelli "interi", che non prevedono numeri dopo la virgola. Sui computer, comunque, l'effettiva gamma di valori dipende dal numero di byte adoperati per conservare i numeri in memoria. Questa decisione spetta a chi realizza il compilatore del linguaggio e tali caratteristiche, chiamate "dipendenti dall'implementazione" devono essere specificate nella documentazione tecnica da sottoporre alla ISO (International Standard Organisation), l'organismo preposto alla verifica della rispondenza di un linguaggio PASCAL agli standard prestabiliti.



La gamma degli interi è da — 32.768 a 32.767 oppure da — 2.147.483.648 a 2.147.483.647, a seconda che vengano adoperati due o quattro byte per memorizzare i valori. Il PASCAL possiede un identificatore costante predefinito, MaxInt, che contiene il massimo valore intero rappresentabile dalla specifica implementazione. Pertanto, tale limite è facilmente individuabile digitando:

```
WriteLn ('MaxInt vale:',MaxInt)
```

Anche i numeri reali hanno una precisa gamma di validità (intorno a 1.7E+38) e di precisione (nei casi peggiori sei o sette cifre significative). La notazione qui usata per esprimere tale limite, conosciuta come "notazione scientifica", è quella solitamente adoperata in PASCAL per i numeri reali, ma è consentita anche la forma più consueta, come in 123.456.
Si rammenti, però, che nei numeri reali occorre sempre interporre un punto decimale (non la virgola) tra la parte intera e quella decimale e che entrambe le parti devono essere presenti. Ad esempio: 0.1 e 1.0E—1 sono numeri reali validi, mentre .1 o 1E—1 non lo sono. Fortunatamente, queste regole ferree si applicano soltanto ai numeri che devono essere riconosciuti come reali all'interno del listato del programma: i valori immessi da tastiera vengono automaticamente convertiti in numeri reali (se questo è il formato previsto), anche se sono semplici interi.

Char è una semplice abbreviazione che indica "carattere" e questo tipo di variabile è previsto per contenere i simboli che fanno parte dell'insieme ASCII. Ai fini di compatibilità, il PASCAL dà per scontato:

- L'ordine alfabetico dei caratteri da A a Z (ossia il codice di A deve essere minore di quello di B e così via)
- L'ordine numerico dei caratteri numerici, da 0 a 9, che deve essere progressivo e contiguo (qualsiasi codice sia attribuito a 0, quello di 1 deve essere immediatamente successivo e maggiore)

Detto per inciso, nel set dei caratteri ASCII l'alfabeto, oltre ad essere ordinato alfabeticamente è anche contiguo, benché ciò non sia essenziale per il PASCAL. Ogni carattere possiede un codice che fa parte di un sottoinsieme dei valori interi: i codici ASCII infatti appartengono alla gamma 0-127 (anche se su alcuni apparecchi questa è estesa fino a 255, comprendendo speciali simboli grafici).
Il PASCAL ha una funzione predefinita, Ord, che fornisce il numero intero estratto dal suo argomento. Ord(A), ad esempio, produce come risultato 65, ossia il codice ASCII di tale lettera. Esiste anche una funzione opposta, chr, che dà il carattere corrispondente ad un codice: chr(65) dà per risultato A (chi è abituato al BASIC noti l'assenza del simbolo $).
La gamma dei valori (sia per i caratteri che per gli interi) è ben definita per ogni implementazione e costituisce una scala ordinata di valori costanti. Ad essi ci si riferisce spesso come tipi "ordinali" o "scalari". Qualunque sia un dato valore, se ne conosce sempre il precedente e il successivo, purché esistano. Essi si ricavano con le funzioni:

```
pred(valore) (per il precedessore)
succ(valore) (per il successore)
```

Pertanto succ(3) corrisponde sempre al carattere di valore 4, ma pred(Z) corrisponde a Y solo in certi set di caratteri, tra i quali l'ASCII. Analogamente, Pred(MaxInt) può essere 32.766 oppure 2.147.483.646. La funzione chr può essere adoperata soltanto con argomenti che siano codici di caratteri. In tutte le altre funzioni scalari si può usare qualsiasi tipo di scalare, benché ord, usata con interi, ha il solo effetto di fornire il valore dell'argomento specificato.
Le variabili Boolean sono il tipo più semplice di variabili scalari, poiché la loro scala è composta da due soli valori: false (falso) o true (vero), in questo ordine. Poiché questi sono semplici valori scalari, ad essi può essere applicata qualsiasi funzione scalare. L'ordinale che corrisponde a false è 0 mentre ord(true) produce 1. Le

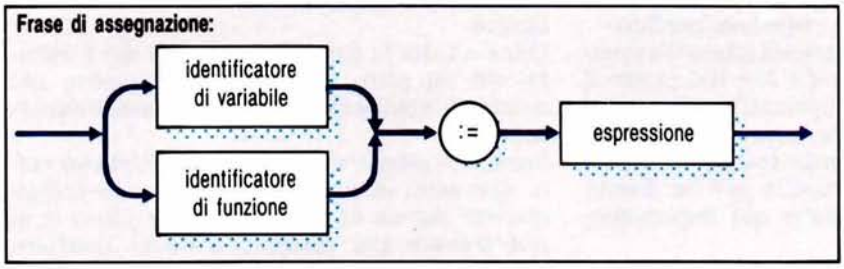
altre funzioni scalari prec e succ, comunque, in questo caso hanno scarso significato.
Il compilatore PASCAL dovrebbe sempre segnalare un errore se si scrive qualcosa come WriteLn(pred(false)), poiché questo è un tentativo di valutare un elemento che non esiste.
Ecco una sezione dichiarativa di costanti di un programma, dove ci sono semplici tipi di ordinali:

```
CONST
    IVA          = 0.18;
    colonne      = 40;
    spazio       = ' ';
    verifica     = false;
```

Eguaglianze e assegnazioni

Il segno uguale, in PASCAL, è adoperato solo per definire l'uguaglianza con valori delle costanti. Nella dichiarazione di variabili (sezione VAR del listato), il carattere che separa l'identificatore dal tipo è il due punti:

```
VAR
    tasso        :real;
    numero       :integer;
    simbolo      :char;
    fatto        :boolean;
```



LIZ DIXON

Quando si desidera assegnare un valore a tali variabili, occorre adoperare il simbolo composto (:=). Questa distinzione serve per riconoscere a colpo d'occhio i tre diversi tipi di operazioni: le definizioni CONST si riferiscono a valori costanti, le dichiarazioni VAR servono a riservare spazio in memoria, mentre le assegnazioni attribuiscono un valore (il più delle volte provvisorio) ad un identificatore.

Istruzioni multiple

Quando due o più istruzioni devono essere eseguite come parte di un singolo processo, occorre racchiuderle tra le due parole riservate BEGIN e END, formando così una frase "multipla". Si rammenti che, all'interno di questa, valgono le solite regole sull'uso del punto e virgola. Ecco un esempio di programma contenente una frase multipla, in cui vengono adoperate gran parte delle funzioni fin qui esaminate. D'ora in poi, per chiarezza, viene adottata la convenzione in base alla quale le parole riservate del PASCAL sono scritte in maiuscole, per distinguerle dagli identificatori. Si noti la spaziatura tra le linee, che serve ad evidenziare ciascun blocco o sezione di programma:

```
PROGRAM Cerchio (input, output);

CONST
    π            = 3.1415926536;
    richiesta    = 'Immettere il raggio:';

VAR
    raggio,
    area         :real;

BEGIN
    WriteLn;
    write (richiesta);
    read (raggio);
    area := π * raggio * raggio;
    WriteLn;
    WriteLn ('Area del cerchio', 'con raggio',raggio : 8 : 3);
    WriteLn ('misura:',area : 10 : 3)
END.
```

Ci sono due aspetti della sintassi che vale la pena di far notare. In primo luogo, nella parte VAR vengono dichiarate due variabili dello stesso tipo (entrambe real). Non occorrono due dichiarazioni separate, ma basta invece elencare i vari elementi separandoli con una virgola. Questa è una regola universale del PASCAL, che vale anche quando si specificano più argomenti per una procedura (si veda ad esempio la penultima WriteLn nel programma).
In secondo luogo si noti il sistema adoperato per formattare l'emissione di valori numerici, che altrimenti verrebbero visualizzati in notazione scientifica: i due numeri preceduti dai "due punti" si riferiscono alla lunghezza in cifre della parte intera e di quella decimale del numero da visualizzare.
Nel programma Cerchio il numero di posti decimali è 3 sia per il raggio che per l'area. Ma siccome la parte intera dell'area è sicuramente composta da un maggior numero di cifre, ad essa sono riservati 10 posti, rispetto agli 8 riservati per il raggio.
Le lunghezze devono essere maggiori di zero e tener conto di un eventuale segno del valore da visualizzare e della posizione del punto decimale. In genere valori di lunghezza illegali generano un errore in fase di compilazione, sebbene in certi casi il numero venga semplicemente visualizzato in notazione scientifica, ignorando cioè il tentato formattamento: WriteLn(X:6:2), ad esempio, non prevede spazio per numeri maggiori di 99.99.
Il valore visualizzato viene automaticamente arrotondato sull'ultima cifra e ciò risulta molto utile. Inoltre, la lunghezza non deve essere necessariamente una costante: può essere una variabile o un'espressione, facilitando così il compito di tabulare una serie di risultati in forma ordinata. Con tutti gli altri tipi di dati è consentito un solo valore per indicarne la lunghezza del campo. Specificando una lunghezza di 1, i numeri interi vengono visualizzati nel minor spazio possibile, senza alcuna spaziatura. Questa è un'arma a doppio taglio, poiché se non si sta attenti, due diversi valori, ad esempio 12 e 34, possono risultare affiancati (1234), provocando un'interpretazione completamente errata.

Un gioco di simulazione

Inizia lo sviluppo pratico di un divertente gioco: eccone la descrizione

Oltre ad offrire la possibilità di accontentare i patiti dei giochi arcade, impegnandoli in vere e proprie gare di prontezza dei riflessi, il computer è un ottimo mezzo per una sfida di carattere ben diverso: i giochi di simulazione. Come la stessa definizione suggerisce, si tratta di programmi nei quali viene riprodotto un modello teorico il più possibile aderente alla realtà. Talvolta questi modelli sono gestiti "in tempo reale" (è il caso di alcuni simulatori di volo, ad esempio) altre volte si tratta di giochi strategici.

Nel genere strategico, al giocatore o al gruppo di giocatori viene assegnata una determinata serie di compiti e sufficienti informazioni per giungere alle decisioni migliori. Un tipico esempio è rappresentato da un gioco basato sul commercio, in cui ai partecipanti viene assegnata in partenza una somma da investire per ottenere il massimo profitto.

La differenza principale tra un gioco d'avventura e uno di simulazione è che nel primo il giocatore affronta vari "pericoli" o situazioni improvvise ed impreviste, mentre nel secondo egli può progettare una tattica di gioco, adoperando le proprie risorse per far fronte alle varie situazioni (anche qui impreviste, ma non sempre).



LIZ HEANEY

Mondi immaginari

L'illustrazione mostra tre esempi rappresentativi dei giochi di simulazione. "Space Shuttle" è una simulazione arcade piuttosto realistica del vero modulo spaziale. "Air Traffic Control" è una simulazione in tempo reale della frenetica attività di un controllore di volo. "The Great Nordic War" è un cosiddetto "wargame", in cui il giocatore assume il ruolo di Carlo XII di Svezia

Questo genere di gioco, quindi, richiede ai partecipanti una ponderata capacità organizzativa e spesso un buon fiuto, piuttosto che abilità nel prendere decisioni immediate e fini a se stesse.

Comunque in una buona simulazione, come del resto nel mondo reale, non c'è mai niente di assolutamente certo: c'è sempre un fattore casuale che può mandare all'aria anche i piani meglio concepiti. Il giocatore deve pertanto sapersi adattare anche a bruschi cambiamenti.

I giochi di simulazione si prestano ad essere giocati anche da gruppi di giocatori, che riuniscono le proprie forze intellettive contro quelle simulate dal computer.

Per questa ragione talvolta l'uso di questo genere di giochi viene consentito (e in qualche caso incoraggiato) anche nell'ambito scolastico.

Oltre a tutte le caratteristiche fin qui elencate, c'è un altro aspetto molto insolito nei giochi di simulazione, ma non meno importante.

Spesso il giocatore viene a confrontarsi con la necessità di prendere decisioni di ordine morale. Ad un certo punto di un gioco ci si può trovare, per esempio, a dover scegliere tra un maggior profitto economico e il benessere dei propri dipendenti. Se al gioco partecipano più persone, dilemmi di questo genere possono favorire uno scambio di opinioni su argomenti che fanno parte della vita reale.

Sono molti gli scenari adatti ad essere usati nei giochi di simulazione, poiché in fondo si tratta di costruire una trama attorno alla semplice assegnazione di un particolare compito. In alcuni casi è addirittura il giocatore che può scegliere il tipo di scenario in cui giocare. Lo scopo del gioco può essere quello di mantenere in vita un'attività commerciale per un tempo prefissato oppure quello di raggiungere un fatturato "minimo" di un miliardo.

Molti giochi sono basati su simulazioni commerciali e finanziarie (vedere a pag. 881), ma alcuni hanno scopi più filantropici.

Un gioco di simulazione che sta avendo un certo successo, anche nell'ambito scolastico, è basato sulla ricerca e il recupero di un galeone elisabettiano, il *Mary Rose*. I giocatori dispongono di materiale documentativo addizionale, oltre alle indicazioni fornite via via sullo schermo.

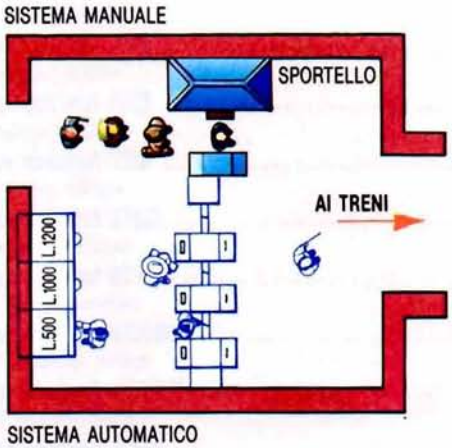
Dopo aver intessuto le lodi dei giochi di simulazione in generale, veniamo al progetto sviluppato nelle prossime parti di questa sezione.

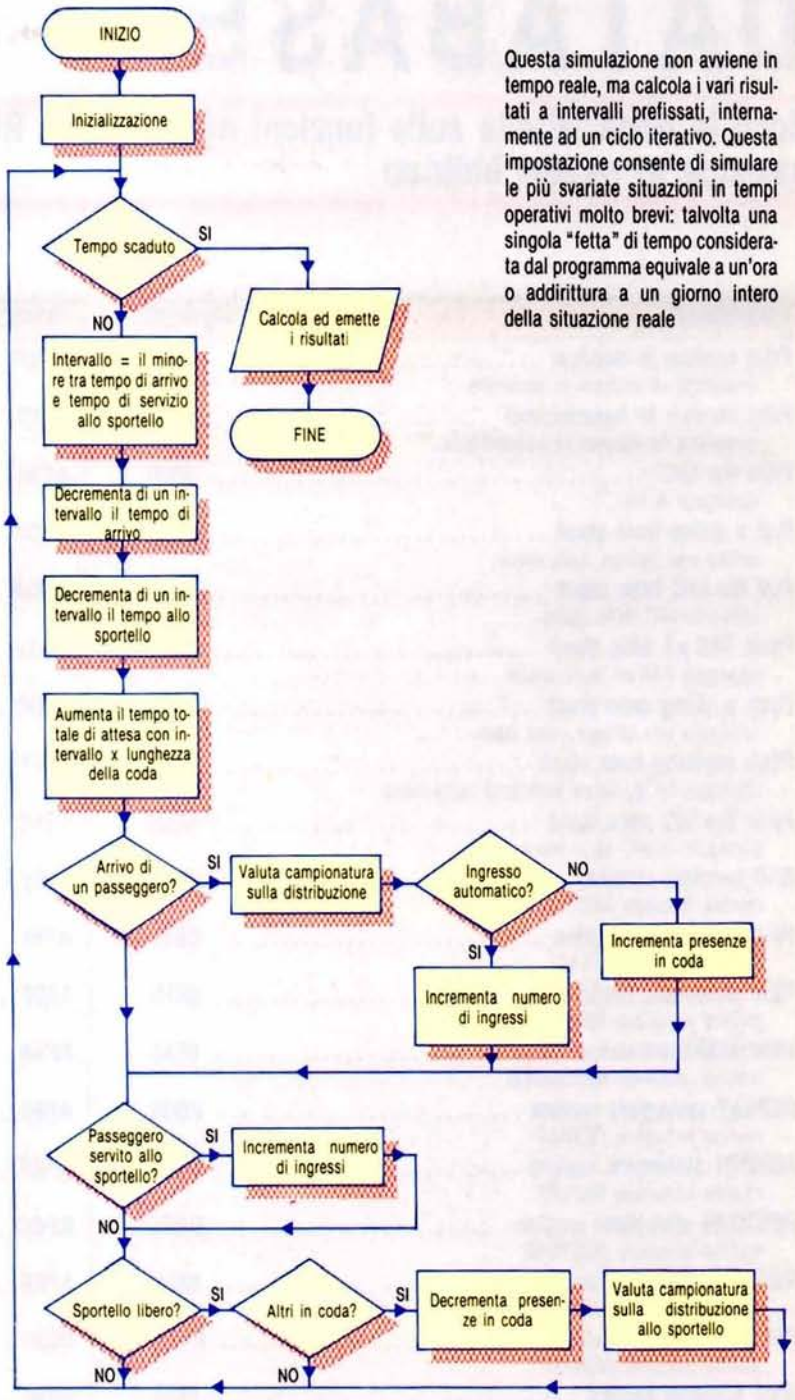
VIAGGIO IN METROPOLITANA

Le simulazioni, oggi diffusamente adoperate sia per l'insegnamento sia per puro e semplice divertimento, vennero inizialmente realizzate su computer come strumenti di pianificazione. Le simulazioni sono di grandissima utilità per valutare, ad esempio, lo scorrimento del traffico su certi percorsi critici o la distribuzione di servizi. In genere questo tipo di simulazioni ricorre a complesse formule per ricavare dati statistici attendibili, da usare come campioni vicini alla realtà.

Esaminiamo un esempio di simulazione relativamente semplice: il sistema di accesso a una metropolitana. Il sistema tradizionale prevede l'acquisto dei biglietti presso l'apposito sportello: quello da simulare, per verifica, prevede invece l'impiego di biglietterie automatiche e di una serie di barriere automatiche per il controllo dei biglietti. Il diagramma di flusso illustra il processo necessario per realizzare la simulazione del nuovo sistema. Le distribuzioni sono tre: quelle dell'affluenza dei viaggiatori, quella relativa alla vendita manuale (allo sportello) dei biglietti e quella al sistema di ingresso automatizzato.

Le prime due distribuzioni sono derivate da osservazioni statistiche e tramite metodi matematici, mentre la terza viene variata ad ogni nuovo lancio del programma, per osservare le diverse ramificazioni e proporzioni coinvolte nella simulazione dei processi manuale e automatico. Questa simulazione intende calcolare il tempo medio di attesa, in coda, dei passeggeri che entrano nella metropolitana





Questa simulazione non avviene in tempo reale, ma calcola i vari risultati a intervalli prefissati, internamente ad un ciclo iterativo. Questa impostazione consente di simulare le più svariate situazioni in tempi operativi molto brevi: talvolta una singola "fetta" di tempo considerata dal programma equivale a un'ora o addirittura a un giorno intero della situazione reale

Si tratta di una simulazione di tipo commerciale, ma ambientata all'incirca nel XVI secolo: il giocatore assume il ruolo del comandante di un vascello che intraprende un viaggio per le lontane Americhe.

Prima di affrontare il viaggio vero e proprio, occorre ingaggiare l'equipaggio ("progettando" bene la sua composizione ed il suo costo) ed acquistare derrate e materiale vario occorrente per la traversata, durante la quale non mancano imprevisti.

Il programma è stato realizzato suddividendolo in una serie di moduli indipendenti, cosicché lo scenario può facilmente essere modificato, pur mantenendo l'ossatura del programma originale.

Chi desidera sbizzarrirsi, può ad esempio trasformarlo in un gioco spaziale, in cui il commercio avviene con pianeti distanti svariati anni-luce.

Scopo del gioco, adatto a uno o più giocatori, è quello di commerciare cercando di accumulare il maggior profitto possibile, non trascurando però il benessere e l'incolumità del proprio equipaggio. Il gioco è suddiviso in tre fasi distinte: la preparazione al viaggio, il viaggio stesso e la contrattazione commerciale.

DATABASE

Ecco la nona tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi

DESCRIZIONE	INDIRIZZO
Print number in decimal visualizza un numero in decimale	**9EDF**
Print number in hexadecimal visualizza un numero in esadecimale	**9E90**
Print the IAC visualizza lo IAC	**991F**
Pull a string from stack estrae una stringa dallo stack	**BDCB**
Pull the IAC from stack estrae lo IAC dallo stack	**BDEA**
Push FAC#1 onto stack appoggia FAC#1 sullo stack	**BD51**
Push a string onto stack appoggia una stringa sullo stack	**BDB2**
Push anything onto stack appoggia un qualsiasi elemento sullo stack	**BD90**
Push the IAC onto stack appoggia lo IAC sullo stack	**BD94**
RAD function routine routine funzione RAD	**ABB1**
READ statement routine routine istruzione READ	**BB1F**
REM statement routine routine istruzione REM	**8B7D**
RENUMBER command routine routine comando RENUMBER	**8FA3**
REPEAT statement routine routine istruzione REPEAT	**BBE4**
REPORT statement routine routine istruzione REPORT	**BFE4**
RESTORE statement routine routine istruzione RESTORE	**BAE6**
RETURN statement routine routine istruzione RETURN	**B8B6**
RIGHT$ function routine routine funzione RIGHT$	**AFEE**
RND function routine routine funzione RND	**AF49**
ROM Entry point indirizzo d'ingresso alla ROM	**8000**
ROM header indirizzo testata della ROM	**8005**
RUN statement routine routine istruzione RUN	**BD11**
Real addition somma di numeri reali	**9C8B**
Real comparison confronto tra numeri reali	**9A50**
Real multiplication moltiplicazione tra numeri reali	**9D20**
Real subtraction sottrazione tra numeri reali	**9CE1**

INDIRIZZO	DESCRIZIONE
ADEC	**Evaluate operand** calcolo operando
AE6D	**Decode hexadecimal number** decodifica numero esadecimale
AEB4	**TIME function routine** routine funzione TIME
AEC0	**PAGE function routine** routine funzione PAGE
AECA	**FALSE function routine** routine funzione FALSE
AED1	**LEN function routine** routine funzione LEN
AEDC	**TOP function routine** routine funzione TOP
AEF7	**COUNT function routine** routine funzione COUNT
AEFC	**LOMEM function routine** routine funzione LOMEM
AF03	**HIMEM function routine** routine funzione HIMEM
AF49	**RND function routine** routine funzione RND
AF9F	**ERL function routine** routine funzione ERL
AFA6	**ERR function routine** routine funzione ERR
AFB9	**GET function routine** routine funzione GET
AFBF	**GET$ function routine** routine funzione GET$
AFCC	**LEFT$ function routine** routine funzione LEFT$
AFEE	**RIGHT$ function routine** routine funzione RIGHT$
B026	**INKEY$ function routine** routine funzione INKEY$
B039	**MID$ function routine** routine funzione MID$
B094	**STR$ function routine** routine funzione STR$
B0C2	**STRING$ function routine** routine funzione STRING$
B112	**Search program for a FN/PROC** cerca FN/PROC in un programma
B195	**FN function routine** routine funzione FN
B32C	**Get the value of a variable** estrae valore di una variabile
B3BD	**CHR$ function routine** routine funzione CHR$
B3C5	**Find ERL** trova linea contenente errore (ERL)

Il CX5M della Yamaha

Il primo home computer dotato anche di vere doti musicali

Il CX5M della Yamaha, conforme allo standard MSX, può elaborare programmi di gioco o di word processing, ma la sua vera vocazione è soprattutto la riproduzione di musica elettronica. Un'apposita presa, situata lateralmente, consente di collegare una tastiera musicale YK-10 o YK-01, mentre una coppia di porte MIDI (Musical Instrument Digital Interface) assicura il controllo, da parte del computer, di qualsiasi sintetizzatore o strumento musicale elettronico (che ormai sono tanti), fornito di un'adeguata interfaccia MIDI. Due piccole prese a microjack, infine, permettono il collegamento di due altoparlanti esterni.

Esteriormente il CX5M assomiglia agli altri apparecchi della gamma MSX. La tastiera alfanumerica, composta da 48 tasti, prevede 10 tasti di controllo tipici dello standard MSX, quali il Backspace per la cancellazione, il tasto Graph per la visualizzazione di caratteri grafici e il tasto Code che, premuto assieme a uno dei tasti alfanumerici, consente di visualizzare caratteri appartenenti ad alfabeti non standard.

Al di sopra della tastiera si trovano 5 tasti funzione, per un totale di 10 funzioni programmabili.

Al momento dell'accensione dell'apparecchio, questi tasti sono automaticamente abbinati ad alcuni comandi e funzioni del BASIC (AUTO, LIST, RUN e via dicendo), secondo lo standard MSX. I quattro tasti per il cursore formano una tastierina separata situata nell'angolo inferiore destro della tastiera. Sopra di questi ci sono altri cinque tasti per altrettanti comandi MSX (INSert, DELete, STOP, ecc.), sormontati a loro volta dalla porta d'innesto per le cartucce dei programmi.

Architettura dell'apparecchio

Sul fianco destro il CX5M presenta due porte per joystick di tipo Atari, mentre sul lato posteriore ci sono: un connettore parallelo a pettine, un'interfaccia per stampante compatibile con apparecchi Centronics, una porta per cassette, una presa per monitor a video composito e un'uscita RF per il televisore, un'uscita monofonica per il suono e la presa d'alimentazione.

Le porte MIDI e l'interfaccia per la tastiera musicale, che fanno dell'apparecchio un eccellente strumento musicale, sono contenute in un box separato, che viene collegato ad un connettore a pettine, situato sotto la base del computer.



CHRIS STEVENS

Un'orchestra in miniatura
È probabile che il CX5M della Yamaha venga venduto più nei negozi di musica che non in quelli di computer. Per sottolineare le sue doti musicali, l'apparecchio viene venduto assieme a una tastiera YK-01 o YK-10, simile a quella di un pianoforte, da collegare a una piccola "scatola nera" che contiene l'interfaccia musicale e che va innestata in un connettore situato sotto la base del computer

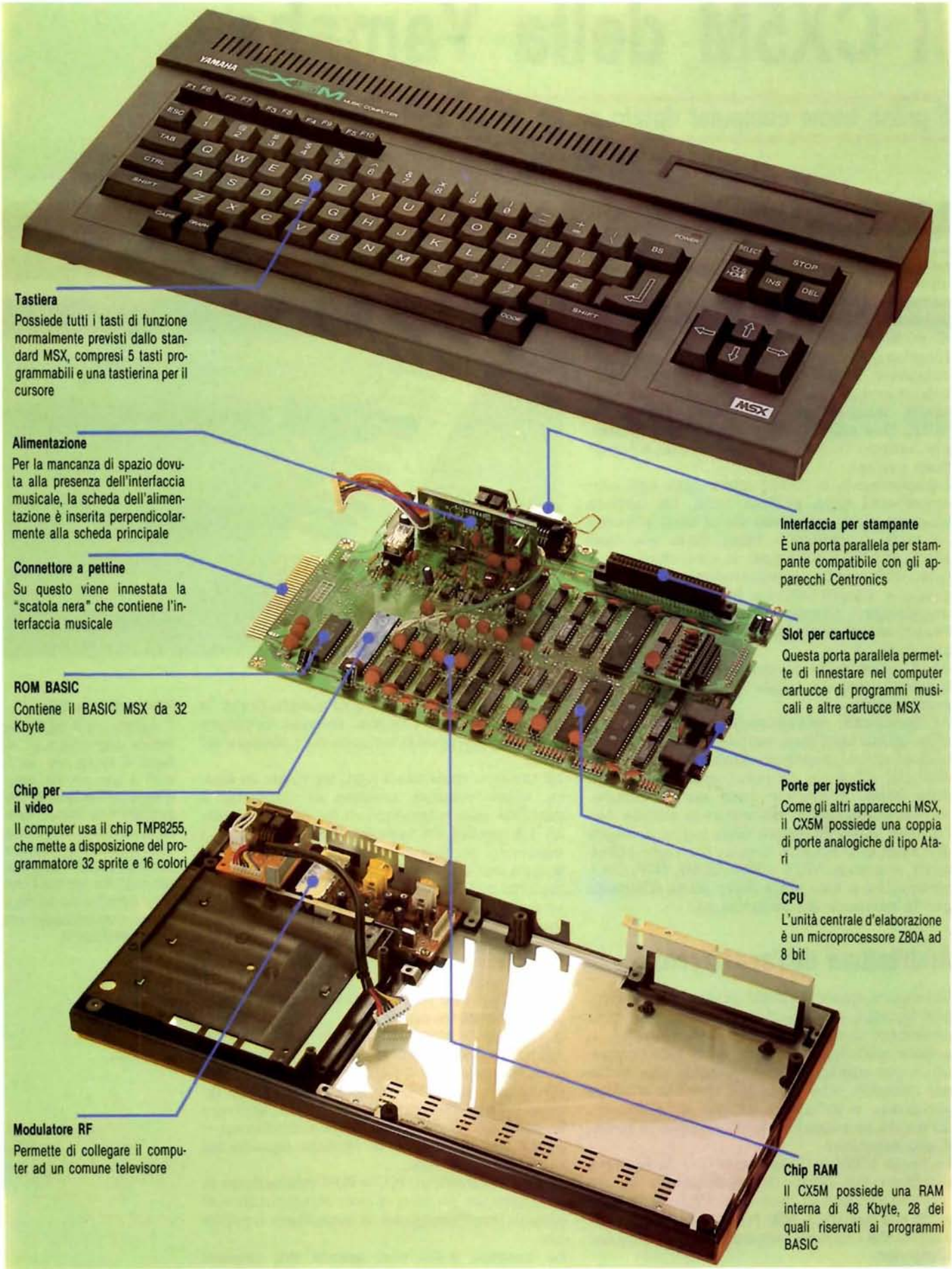
L'inconsueta disposizione di questa porta fa pensare che la Yamaha intenda produrre tutta una serie di interfacce da collegare ad essa.

La tastiera musicale YK-01, mostrata in figura, viene venduta assieme al computer e possiede una estensione di tre ottave e mezzo. La qualità dei tasti è discreta, anche se i tastieristi professionisti li giudicano un po' troppo piccoli. Grazie al software interno, la tastiera può essere programmata per generare contemporaneamente due diverse "voci": una di accompagnamento (sulla parte sinistra della tastiera) e una per la melodia (sul resto della tastiera). Lo strumento può produrre suoni monofonici o polifonici (o di entrambi i tipi) ed emettere fino a otto note contemporaneamente.

Il video ha il tipico colore blu degli schermi MSX. Per selezionare il programma musicale occorre impartire il comando CALL MUSIC, che visualizza un menu composto da cinque diversi gruppi di opzioni. L'utente può scorrere la lista usando il tasto Return e modificare i parametri delle diverse opzioni agendo sui tasti per il cursore.

I gruppi identificati POLY e MONO permettono di scegliere tra 46 diverse voci strumentali e di selezionare l'emissione in monofonia o polifonia.

La gamma delle voci spazia dai comuni



Tastiera
Possiede tutti i tasti di funzione normalmente previsti dallo standard MSX, compresi 5 tasti programmabili e una tastierina per il cursore

Alimentazione
Per la mancanza di spazio dovuta alla presenza dell'interfaccia musicale, la scheda dell'alimentazione è inserita perpendicolarmente alla scheda principale

Connettore a pettine
Su questo viene innestata la "scatola nera" che contiene l'interfaccia musicale

ROM BASIC
Contiene il BASIC MSX da 32 Kbyte

Chip per il video
Il computer usa il chip TMP8255, che mette a disposizione del programmatore 32 sprite e 16 colori

Modulatore RF
Permette di collegare il computer ad un comune televisore

Interfaccia per stampante
È una porta parallela per stampante compatibile con gli apparecchi Centronics

Slot per cartucce
Questa porta parallela permette di innestare nel computer cartucce di programmi musicali e altre cartucce MSX

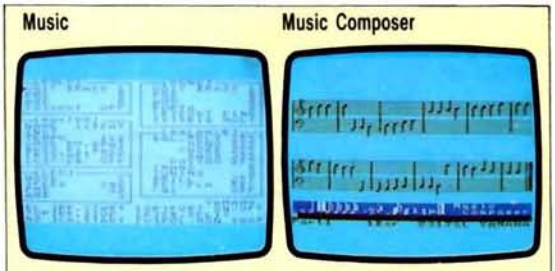
Porte per joystick
Come gli altri apparecchi MSX, il CX5M possiede una coppia di porte analogiche di tipo Atari

CPU
L'unità centrale d'elaborazione è un microprocessore Z80A ad 8 bit

Chip RAM
Il CX5M possiede una RAM interna di 48 Kbyte, 28 dei quali riservati ai programmi BASIC



Queste sono alcune delle cartucce prodotte per il CX5M. I diversi programmi in esse contenuti permettono al computer di svolgere una particolare funzione: dalla programmazione del sintetizzatore DX-7 ai programmi di composizione musicale. Ogni cartuccia è corredata da un manuale. Il loro costo è abbastanza contenuto, ma l'utente può rimandarne l'acquisto in attesa, magari, di ulteriori riduzioni nel prezzo o di miglioramenti del software



Le immagini sono tratte da due pacchetti del CX5M Yamaha. Music è un programma interno al computer, selezionabile con il comando CALL MUSIC. Dopo aver posizionato il cursore tramite i tasti Function e Return, l'utente può modificare i parametri sullo schermo premendo i tasti per il cursore. Il programma Music Composer permette invece di comporre un brano musicale sul classico pentagramma, visualizzato sullo schermo, e di inviarlo alla stampante per ottenerne una copia su carta

IAN McKINNELL

strumenti musicali (organo, chitarra e strumenti a percussione quali vibrafono e campane) ad una serie di suoni tratti dalla vita reale (ad esempio, la sirena di un'ambulanza e il rumore della pioggia). A causa della loro particolare natura, alcune voci non possono essere usate assieme alla funzione di "prolungamento" del suono. Modificando le voci dei modi POLY e MONO, l'utente può suddividere la tastiera in due sezioni, una monofonica e l'altra polifonica, assegnando ai diversi tasti tonalità contrastanti.

Naturalmente la tastiera può essere usata anche in un solo modo (monofonico o polifonico) mentre non è possibile suddividerla in due sezioni POLY.

I suoni prodotti hanno un'ottima definizione, anche se emessi dall'altoparlante di un comune televisore, ma alcuni corrispondono solo vagamente al loro nome. Ad esempio, il rumore del treno ricorda più il suono di un comune organo elettrico che lo sferragliare di un treno.

Le opzioni RHYTHM del menu permettono di combinare tra loro accordi, bassi e percussioni e di produrre sei diversi tipi di accompagnamento ritmico. Le note dei bassi e degli accordi possono essere programmate dall'utente, che può modificare anche la cadenza del ritmo. Un'opzione del menu consente di variare la tonalità del ritmo direttamente dalla tastiera musicale. Modificando la forma dell'onda sonora con il gruppo di opzioni LFO (Low Frequency Oscillator), che governano l'oscillatore di bassa frequenza, è possibile modulare le tonalità ottenendo effetti sonori molto originali. Le opzioni BALANCE consentono invece di definire il volume di ciascun elemento ritmico e combinandole ai modi MONO e POLY, è possibile suonare in sottofondo l'accompagnamento ritmico MONO ed alzare il volume dei suoni POLY, accrescendo l'incisività del brano suonato. Una volta programmato per intero un pezzo musicale, questo può essere salvato su cassetta e richiamato in seguito.

La Yamaha ha prodotto una serie di cartucce musicali per il CX5M, indipendenti dal software interno alla tastiera. Ad esempio, la cartuccia *FM Voicing Program* contiene una versione estesa del software interno che consente all'utente una manipolazione molto più sofisticata dei suoni, modificando le frequenze e gli algoritmi che li generano.

Un secondo programma su cartuccia permette al computer di programmare il sintetizzatore di alta qualità DX7, sempre della Yamaha. La cartuccia *FM Music Macro* fornisce al CX5M una serie di comandi BASIC extra, mentre quella chiamata *FM Music Composer* visualizza un pentagramma sul quale l'utente può scrivere un brano musicale in notazione tradizionale e inviarlo ad una stampante esterna.

Benché il software musicale sviluppato dalla Yamaha per il CX5M sia abbastanza completo, si ha l'impressione che i diversi programmi non riescano a sfruttare del tutto le potenzialità dell'apparecchio. I più limitati sembrano essere i programmi di modulazione: è infatti estremamente difficile percepire cambiamenti significativi nel suono di una nota.

Questo difetto è piuttosto inspiegabile, dato che il sistema operativo del CX5M deriva da quello del sintetizzatore DX7 che, in confronto, offre una gamma di suoni molto più estesa.

Inoltre la tecnica adottata per modificare i parametri, pur producendo risultati soddisfacenti, spesso si dimostra scomoda. La modifica dei parametri, ad esempio, avviene tramite i tasti del cursore, mentre lo spostamento del cursore per la selezione delle opzioni è controllato dal tasto Return.

Il CX5M, dunque, è un apparecchio chiaramente destinato agli utenti di home computer appassionati di musica elettronica. In tale prospettiva, la completa dotazione di interfacce MIDI e la tastiera musicale lo rendono assai più interessante di altre macchine di pari prezzo.

Tuttavia proprio questa sua "specializzazione" verso il campo musicale può limitarne il successo, poiché è molto probabile che gli utenti non interessati alla musica elettronica rivolgano la loro attenzione verso altre macchine MSX dal costo inferiore.

YAMAHA CX5M

DIMENSIONI

mm. 413 x 216 x 64

CPU

Z80A, frequenza di lavoro 3,58 MHz

MEMORIA

48 Kbyte di RAM (di cui 24 Kbyte per i programmi BASIC)

VIDEO

40 x di 24 caratteri in modo testuale, 256 x 192 pixel nel modo grafico, con 16 colori e 32 sprite

INTERFACCE

Stampanti tipo Centronics, TV, monitor a video composito. Uscita audio mono e stereo, 2 porte joystick, porta per registratore a cassette, porta per cartucce ROM, bus d'espansione e interfaccia per tastiera musicale. Porte di ingresso e uscita MIDI

LINGUAGGI FORNITI

BASIC, PASCAL e Assembly

TASTIERA

Tipo QWERTY con 67 tasti, tastierina per il cursore, 5 tasti di funzione programmabili. La tastiera musicale è polifonica (fino a 8 note simultanee) con un'estensione di 3,5 ottave

DOCUMENTAZIONE

Il manuale lascia un po' a desiderare: le informalzioni per l'impiego del software musicale sono appena sufficienti e non è prevista alcuna forma di insegnamento passo per passo. Il BASIC MSX è raramente menzionato

PRO

Questo apparecchio è un vero affare per gli appassionati di musica elettronica. Come strumento musicale, infatti, a parità di prezzo il CX5M non ha concorrenti

CONTRO

Il software disponibile può essere indubbiamente migliorato in modo da sfruttare appieno le grandi potenzialità dell'apparecchio. Il manuale è molto limitato. Per chi non fosse affatto interessato all'aspetto musicale è un computer costoso

Leggere la Bibbia

Un database letterario che apre un nuovo orizzonte di studio

"The Word" Processor: Per apparecchi MS-DOS

Distributore: Access Software Ltd, (GB)

Autori: Bible Research Systems, Austin, Texas (USA)

Formato: Dischetti

Alcuni tipi di ricerche letterarie e di studi scolastici implicano una indagine metodica su testi molto vasti. Chi studia la Bibbia, ad esempio, ha bisogno di indici di riferimento specializzati per analizzare i 39 volumi del Vecchio e i 27 del Nuovo Testamento, per un totale di circa 750.000 parole.
Sebbene i computer possano fornire un valido aiuto nell'esame di opere così ampie, l'impiego dei micro in questo campo è spesso condizionato dalle ridotte dimensioni della memoria e dalla limitata capacità dei dischetti. Tuttavia esiste un programma, scritto appositamente, che può facilitare l'analisi dei testi biblici: *"The Word" Processor*, che non solo consente di individuare rapidamente particolari citazioni, ma anche di creare indici incrociati, che riducono sensibilmente i tempi della ricerca.
La completezza del pacchetto mette in luce l'immensa mole di lavoro svolta dai programmatori del *"The Word" Processor*. Il manuale è un agile opuscolo di 36 pagine, ma l'intero programma, compresi tutti i file di testo della Bibbia, è composto da sette dischi a doppia faccia che, come sempre, conviene copiare su dischetti "da lavoro". Se però il drive del proprio computer usa dischetti a singola faccia da 40 tracce, occorrono ben 14 dischetti per contenere l'intera opera, il che dimostra l'inadeguatezza dei dischi da 5 1/4 pollici per le applicazioni di questo genere.

Come funziona "The Word"

Il pacchetto richiede un'accurata installazione. Non è detto che tutti gli studiosi abbiano una profonda pratica di computer, nonostante ciò il funzionamento del programma BASIC è piuttosto complicato e talvolta risponde ai comandi solo sotto PC-DOS e non sotto MS-DOS. L'interprete BASIC e le tracce di sistema dell'apparecchio contenute nel disco del DOS vanno copiate sul disco di lavoro. A questo punto, occorre scegliere se eseguire il programma principale (TWP), adatto al computer IBM PC, oppure la versione (ATWP), per IBM-compatibili, o, infine, la versione speciale (TWP64), per apparecchi con solo 64 Kbyte di memoria. Abitualmente un dischetto contenente le tracce di sistema, il BASIC e i programmi principali è in grado di autocaricarsi in memoria ma, nel caso del *"The Word" Processor*, manca il programma AUTOEXEC.BAT che esegue questa operazione.
Dopo il caricamento e il lancio del programma, sullo schermo compare la pagina d'apertura, dalla quale l'utente può intuire che il pacchetto è decisamente "particolare". Infatti, oltre a presentare l'abituale messaggio di protezione del software («Questo software è protetto dalle leggi USA sui diritti d'autore. La riproduzione non autorizzata è punibile con il carcere o con un'ammenda massima di 10.000 dollari»), la prima pagina contiene la seguente citazione biblica:

EXODUS 20:15 "Thou shalt not steal."

il cui significato non lascia dubbi («Non rubare!»). Questo perentorio messaggio è seguito da un menu d'apertura, in cui ogni opzione viene selezionata agendo sui tasti funzione.
Premendo il tasto di funzione <F2>, ad esempio, viene visualizzato un sub-menu contenente 9 opzioni, tra le quali quella per fissare la larghezza dello schermo (40 o 80 caratteri), per i margini destro e sinistro sulla stampante, per la spaziatura tra le righe, per il numero di righe in ogni pagina, per le dimensioni massime dell'indice, per la visualizzazione di una riga alla volta o di un'intera pagina di testo, per visualizzare il nome di tutti gli indici presenti sul dischetto in uso.
L'opzione <F4>, invece, controlla la gestione su disco dell'intera opera e dei numerosi dischetti di testo. Usando codici a tre lettere, riferiti ai diversi volumi della Bibbia (da Gen a Rev), e i riferimenti ai capitoli e ai versi, l'utente può definire il punto iniziale e finale del parametro di ricerca. Se invece desidera leggere un intero volume, deve omettere i riferimenti al capitolo e al verso: in tal caso il programma considera l'inizio del primo capitolo e la fine dell'ultimo come parametri iniziale e finale del testo da visualizzare.
Dopo aver inserito nel drive un particolare dischetto, è possibile scorrere un intero capitolo, visualizzando il primo capoverso con l'opzione <F10> e muovendosi avanti e indietro con i tasti funzione la cui azione è descritta sull'ultima riga dello schermo. Esiste un'opzione di controllo, che cambia le modalità di scorrimento del testo: da una sola riga a un'intera pagina alla volta, permettendo di leggere un particolare versetto inserito nel contesto della Scrittura.
L'opzione <F2> abilita la funzione SCAN, che permette di definire i criteri di una ricerca. Come avviene nella maggior parte dei database, è possibile individuare tutte le ricorrenze di un particolare gruppo di lettere, anche se situato all'interno di una parola. La procedura di ricerca segnala, ad esempio, le tre

lettere "man" (uomo) interne alla parola "demand" (domanda). Altrimenti si può limitare la ricerca ad una specifica parola o estenderla a tutte quelle che iniziano o terminano con una data sequenza di lettere. I risultati della ricerca vengono poi visualizzati uno alla volta, da soli o all'interno del contesto da cui provengono, a seconda del comando impartito.

L'opzione <F6> permette all'utente di creare indici automaticamente o manualmente. Questi possono essere modificati (ad esempio cancellando un riferimento indesiderato) o combinati tra loro, ricordando però che il numero dei riferimenti di un indice non deve superare 1.020. Questo limite, comunque, può essere portato a 53.040 tramite l'opzione <F2> del menu principale. Gli indici con più di 1.020 riferimenti possono essere archiviati su disco in sezioni separate, per mezzo del submenu OPZIONI DI CONTROLLO. In tal caso la visualizzazione dell'indice subisce una breve interruzione dopo la comparsa del riferimento 1020, perché il programma deve caricare in memoria, leggendola dal dischetto, la sezione successiva. Infine l'utente può creare manualmente un indice degli argomenti trattati in ciascun versetto, indicando il titolo dell'argomento e il numero del versetto corrispondente.

A parte le notevoli difficoltà di elaborazione, dovute alla vastità della Bibbia, il principale problema che si incontra lavorando col *"The Word" Processor* dipende dall'edizione presa in esame, ossia quella di King James, del 1611.

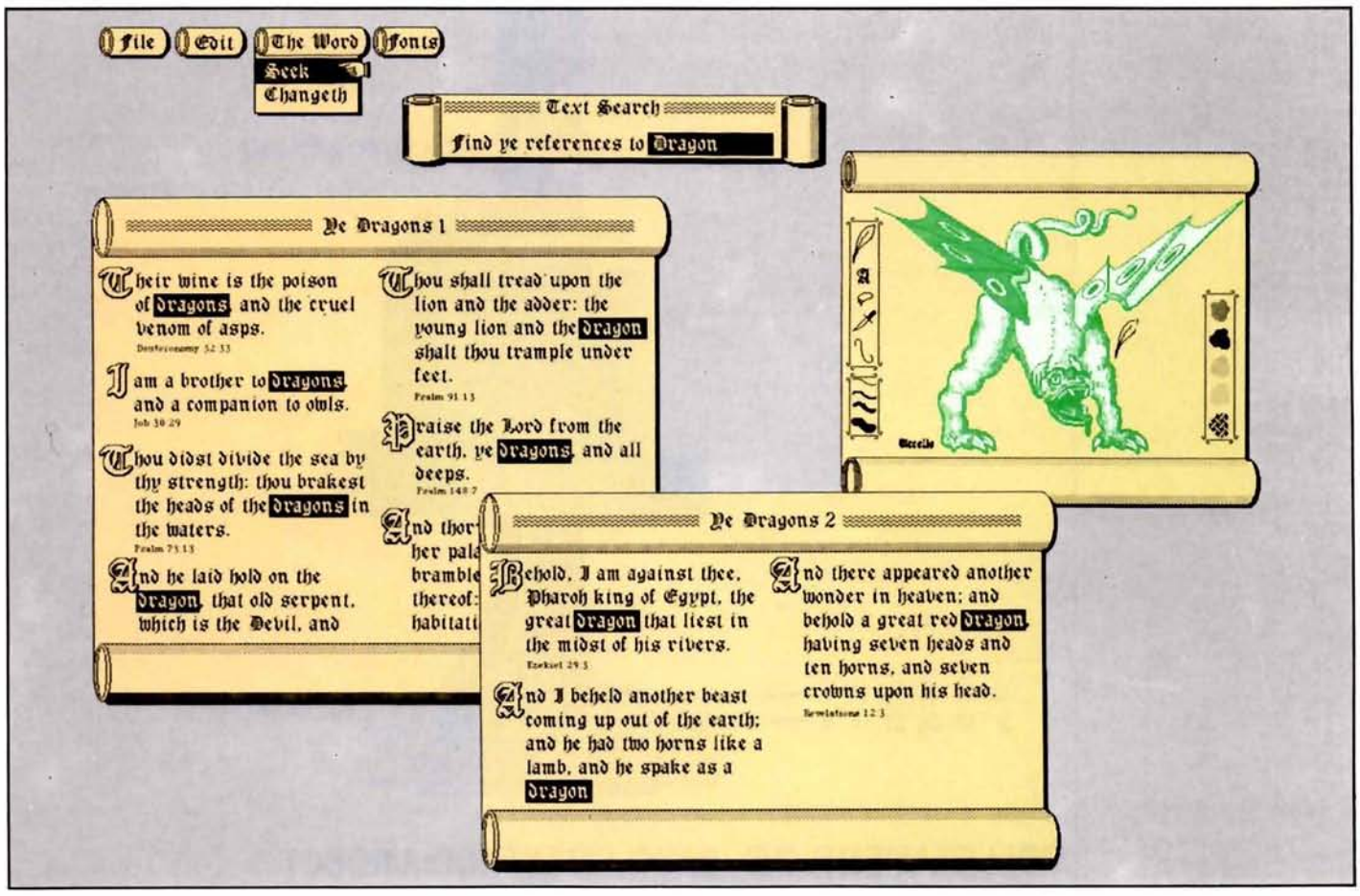
La traduzione inglese contiene almeno trecento parole e frasi che hanno cambiato significato nei tre secoli successivi e anche alcuni errori di traduzione.

Fortunatamente, al pacchetto può essere abbinato uno speciale programma "traslitteratore" dal greco: il *"The Greek Transliterator"*, che permette di eliminare parte degli errori, almeno per quanto riguarda il Nuovo Testamento. Il programma consente di individuare e visualizzare la corrispondente parola del testo greco originale (scritto fortunatamente in caratteri romani!) e di interpretarne correttamente il significato. Inoltre può consultare gli indici creati con il *TWP* e crearne di propri contenenti parole greche. Il prezzo di questo programma è pari a quello del *TWP*.

Attualmente manca ancora un programma di traslitterazione dall'ebraico per il Vecchio Testamento, sebbene a Gerusalemme sia stato installato un elaboratore dotato di un database di pubblico accesso per lo studio del Talmud e di altri testi sacri ebraici. Probabilmente per lo studio computerizzato della Bibbia un simile sistema sarebbe da preferire ai limitati database individuali per microcomputer, tuttavia le tecniche impiegate per il *"The Word" Processor* mantengono una loro validità e potrebbero essere usate anche per altre applicazioni di pari respiro, come i drammi di Shakespeare o gli scritti di Marx ed Engels.

King James e il drago

Quando l'utente digita sulla tastiera la parola "dragon" (drago), il "The Word" Processor visualizza gli indici di riferimento mostrati nella figura e altre informazioni, semplificando notevolmente la ricerca e lo studio dei testi biblici e dei loro significati. L'edizione della Bibbia presa in esame risale al 1611 ed è quella di King James



IAN McKINNELL

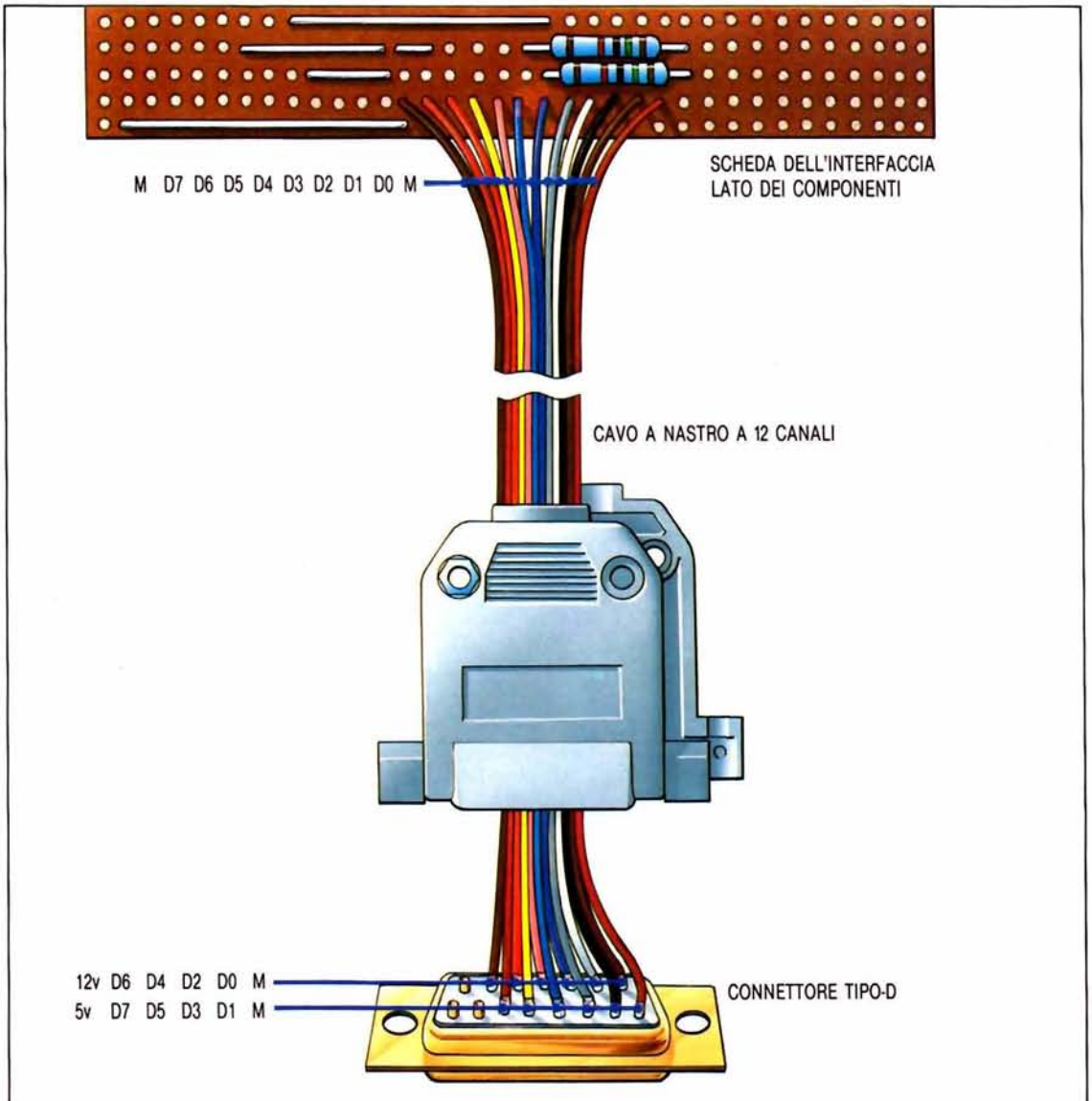
Collegare l'interfaccia

Realizziamo il cavo di collegamento per lo Spectrum

La costruzione dell'interfaccia si conclude saldando il cavo a nastro a 12 canali collegato alla scheda dell'interfaccia al connettore tipo-D da innestare sul robot. Le otto linee per i dati provenienti dalla porta di I/O dello Spectrum vengono rese disponibili sul bus dei dati attraverso i circuiti dell'interfaccia, mentre le linee d'alimentazione (12 e 5 volt) provengono direttamente dalla porta d'espansione dello Spectrum. Non occorre, quindi, nessuna alimentazione esterna per azionare il robot, poiché sia per il computer che per il robot basta quella fornita dallo Spectrum.
Il cavo di collegamento va preparato asportando quattro fili dal cavo a nastro indicato nell'elenco dei componenti (vedere a pag. 1492). Entrambe le estremità dei dodici cavetti rimasti vanno spellate e stagnate, per poi essere saldate da un lato alla scheda dell'interfaccia e dall'altro al connettore tipo-D, attenendosi allo schema in figura. Il lavoro eseguito va controllato attentamente, assicurandosi in particolare che le linee d'alimentazione da 12 e 5 volt siano collegate ai rispettivi piedini del connettore.
Adesso l'interfaccia è completa e può essere inserita nella porta d'espansione sul retro dello Spectrum, prima di dare corrente e di verifi-

Collegamento del cavo a 12 canali

Lo schema del collegamento tra il connettore tipo-D e la scheda d'interfaccia è piuttosto semplice. I fili vengono saldati alternativamente ai piedini superiori e a quelli inferiori del connettore



COLLEGAMENTI DEL CAVO INTERFACCIA/ROBOT

carne il funzionamento. La scheda deve essere innestata con il lato dei componenti rivolto verso l'alto. Se il "blocco" è stato inserito nella posizione 5 del connettore a pettine della porta d'espansione, la scheda non può essere montata in altro modo. Qualora l'innesto risulti difficoltoso, si faccia oscillare leggermente la scheda a destra e a sinistra per facilitare lo scorrimento del connettore a pettine nella fessura della porta d'espansione. Adesso si colleghi il cavo a nastro al robot e si accenda il sistema. Se tutto funziona, sullo schermo appare la scritta di "copyright" della Sinclair. Ora possiamo lanciare un programma di collaudo.

Controllo del motore

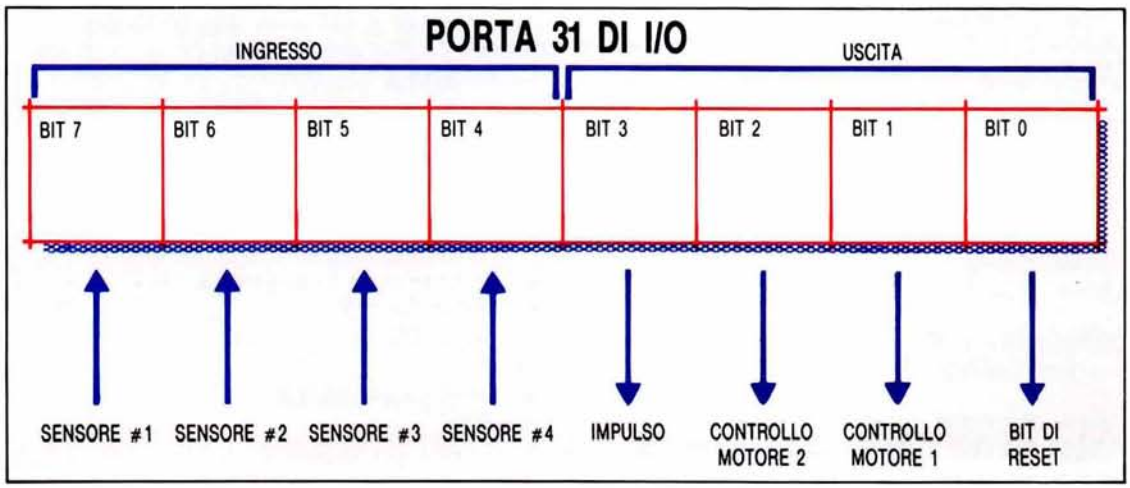
Gli otto bit sulla porta 31 di I/O del computer vengono usati per controllare i motori e ricevere i segnali dai sensori del robot. I quattro bit più bassi controllano il movimento: i bit 1 e 2 controllano il senso di rotazione dei due motori passo passo montati sul robot. Le quattro direzioni vengono selezionate assegnando ai due bit diverse combinazioni di valori. Il bit 3 è il bit che governa gli impulsi. Quando è posto a 1, entrambi i motori ruotano di un passo nella direzione indicata dai due bit precedenti. Il bit 0 è quello di reset e in condizioni normali contiene il valore 1.
Per collaudare del circuito d'uscita dell'interfaccia si può usare il programma di controllo del Robot, che permette di pilotarlo usando la tastiera. Le lettere T, B, F e H corrispondono rispettivamente a spostamenti in avanti, indietro, a sinistra e a destra. La routine di inizializzazione definisce 4 variabili relative alle quattro direzioni del robot. I loro valori binari corrispondono alle quattro possibili combinazioni dei bit 1 e 2 sulla porta 31 di I/O:

Direzione	Variabile	Valore decimale	Valore binario
Avanti	av	4	0100
Indietro	nd	2	0010
Sinistra	sx	6	0110
Destra	dx	0	0000

La variabile dr contiene la direzione del robot. Per muovere il robot in una particolare direzione occorre inviare un impulso ai motori ponendo alternativamente ad 1 e a 0 il bit 3. Questo compito è affidato alla subroutine alla linea 2000, che genera l'impulso: questa usa l'istruzione OUT per inviare un valore alla porta di I/O 31 (OUT funziona in modo simile a POKE). Il valore viene calcolato sommando alla variabile dr il valore 8, che pone il bit 3 ad 1, e il valore 1, che pone il bit 1 a 0: l'operazione eseguita è quindi dr+9. Per azzerare il bit 3 basta eliminare il valore 8 e sommare semplicemente dr+1. L'accensione (1) e il successivo spegnimento (0) del bit 3 fanno ruotare i motori di un passo, ossia di 7,5°. Poiché questo angolo corrisponde ad una rotazione della ruota di circa 1 mm., è necessario inserire la procedura che genera il segnale in un ciclo, ripetuto per un numero di volte indicato dalla variabile m.
Per cambiare la direzione del robot basta premere uno dei tasti di controllo che modificano il valore di dr intervenendo sui bit direzionali.

Controllo dell'input

I quattro bit più alti, ciascuno dei quali corrisponde a una delle prese volanti poste sul coperchio del robot, vengono usati per intercettare i segnali generati dai sensori. In condizioni normali questi bit valgono 1, a meno che non vengano messi a massa dalla chiusura di uno degli interruttori: in tal caso il bit corrispondente assume il valore 0. Per intercettare i segnali in ingresso relativi ad ogni bit, è necessario isolare ciascuno dei quattro bit più alti e verificarne il contenuto. Nella maggior parte dei dialetti BASIC questa operazione viene eseguita dall'istruzione logica AND, che permette di mascherare i bit non sottoposti a verifica. Purtroppo l'istruzione AND, nel BASIC dello Spectrum, non è abbastanza sofisticata da consentire il mascheramento selettivo (individuale) dei bit di un numero. Tuttavia il processore Z80 prevede un'istruzione AND capace di svolgere questa operazione. Possiamo quindi scrivere una semplice routine in codice macchina



Configurazione della porta
L'interfaccia dello Spectrum seleziona per l'uscita i quattro bit più bassi e per l'ingresso i quattro più alti della porta 31 di I/O. I primi controllano il funzionamento dei motori passo passo, i secondi intercettano i segnali generati dai sensori

che ponga in AND due numeri e calcoli il risultato. Per eseguire la routine è necessario depositare (con POKE) in due locazioni di memoria i due numeri sui quali vogliamo operare e usare quindi l'istruzione USR per riportare il risultato in BASIC. Questo metodo indiretto consente la lettura indipendente di ciascuna linea in ingresso.
Il programma *Paraurti* per lo Spectrum verifica il funzionamento dei circuiti d'ingresso dell'interfaccia: sposta il robot in avanti finché non urta un oggetto, quindi lo riporta nella posizione di partenza (vedere a pag. 1375).
Le informazioni contenute nel riquadro "Basic & Basic" consentono agli utenti Spectrum di modificare alcuni dei precedenti programmi per il robot. Chi possiede un Commodore 64 deve invece consultare le pagine indicate nel riquadro. Si rammenti, infine, che nelle versioni Spectrum si possono sostituire i nomi di variabile scritti in minuscolo con quelli maiuscoli riportati nelle versioni Commodore.

PROGRAMMA DI CONTROLLO DEL ROBOT

```
  10 REM *** controllo robot - spectrum ***
  20 GO SUB 1000: REM initializza
  30 LET a$=INKEY$: IF a$<>"x" THEN GO SUB 3000:
     REM legge tasto
  40 LET m=10: GO SUB 2000: REM impulso
  50 IF a$<>"x" THEN GO TO 30
  60 OUT 31,0
  70 STOP
  80 :
1000 REM *** inizializza ***
1010 LET av=4: LET nd=2: LET sn=6: LET ds=0
1020 LET dr=av
1030 RETURN
1040 :
2000 REM *** impulso ***
2010 FOR c=1 TO m
2020 OUT 31,dr+9
2030 OUT 31,dr+1
2040 NEXT c
2050 RETURN
2060 :
3000 REM *** legge tasti ***
3010 IF a$="t" THEN LET dr=av: RETURN
3020 IF a$="b" THEN LET dr=nd: RETURN
3030 IF a$="f" THEN LET dr=sn: RETURN
3040 IF a$="h" THEN LET dr=ds: RETURN
3050 RETURN
```

PROGRAMMA PARAURTI PER LO SPECTRUM

```
  10 REM *** respingenti - spectrum ***
  15 CLEAR 32499: LET st=32500
  20 GO SUB 500: REM inizializza
  25 GO SUB 3000: REM carica codice macchina
  30 LET dr=av
  40 REM *** impulso in avanti ***
  50 LET cc=0
  60 GO SUB 1000: LET cc=cc+1: REM impulso
  70 REM ** verifica respingenti **
  80 LET nm=192: GO SUB 2000: REM esegue AND
  90 IF USR st=192 THEN GO TO 60
 100 REM **** torna all'inizio ****
 110 LET dr=nd
 120 FOR i=1 TO cc
 130 GO SUB 1000: REM impulso
 140 NEXT i
 150 OUT 31,0: STOP
 500 **** inizializza ****
 510 LET av=4: LET nd=2: LET lf=6: LET ds=0
 520 RETURN
1000 REM **** impulso ****
1010 OUT 31,dr+9
1020 OUT 31,dr+1
1030 RETURN
2000 REM **** esegue AND ****
2010 POKE st+1,IN 31
2020 POKE st+3,nm: RETURN
3000 REM **** caricatore in codice macchina ****
3010 FOR i=st TO st+8
3020 READ a: POKE i,a
3030 NEXT i
3040 DATA 62,0,14,0,161,6,0,79,201
3050 RETURN
```

Basic & Basic

Cancellare le linee 20, 30 e 70 del programma "Calibrazione Lineare" a pag. 1374 e sostituirle con le seguenti:

```
 80 LET dr = av
100 LET a$ = INKEY$:IF a$ = THEN GO TO 100
280 OUT 31,dr + 9
290 OUT 31,dr + 1
```

Nel programma "Calibrazione Angolare" a pag. 1375, eliminare le linee 20, 30 e 50 e sostituirle con le seguenti:

```
 60 LET dr = de
 80 LET dr = av
100 OUT 31,dr + 9
110 OUT 31,dr + 1
```

Sostituire alle linee 1010, 1020, 1030, 7010 e 7510 del programma "Misurazione con il Robot" a pag. 1425 le linee seguenti:

```
2050 GO SUB 8100:IF USR st<>lb THEN GO TO 2040
2090 GO SUB 8100: IF USR st<>lb THEN GO TO 2080
2150  GO SUB 8100: IF USR st<>rb THEN GO TO 2140
2190  GO SUB 8100: IF USR st<>rb THEN GO TO 2180
2200 OUT 31,0
3010 CLS
3520 GO SUB 8100: IF USR st = nb THEN GO TO 3510
4010 GO SUB 8100: IF USR st = rb THEN LET ss = rt: GO SUB
     5000: RETURN
4020 GO SUB 8100: IF USR st = lb THEN LET ss = si: GO SUB
     5000: RETURN
5020 GO SUB 8100: IF USR st = nb THEN GO TO 5010
6080 GO SUB 8100: IF USR st = nb THEN GO TO 6070
8100 REM **** esegue una AND ****
8110 POKE st + 1,192:POKE st + 3, IN 31: RETURN
8200 REM **** carica il cod. macchina ****
8210 FOR i = st TO st + 8
8220 READ a: POKE i,a
8230 NEXT i
8240 DATA 62,0,14,0,161,6,0,79,201
8250 RETURN
```

Il nuovo e il vecchio

La prima di due parti sull'interazione tra il BASIC e il SO del BBC

La ROM del BASIC, nel Micro BBC, occupa l'area di memoria compresa tra gli indirizzi &8000 e &BFFF (consultare la mappa di memoria a pag. 1359). Questo spazio viene anche detto "area della ROM impaginata". Grazie ad un ragionato impiego delle tecniche di indirizzamento a pagine, numerosi chip ROM di vario tipo possono risiedere contemporaneamente all'interno della macchina e usare lo stesso blocco di memoria. Tuttavia le diverse ROM possono essere attivate, o "impaginate", ma solo una alla volta. I chip DFS ed Econet, i package come il *Wordwise*, il *View* o il *Doctor Disk* e altri linguaggi come il FORTH e il BCPL sono solo alcuni esempi di ROM che occupano quest'area di memoria. Il principale vantaggio offerto dal loro impiego è, ovviamente, il risparmio di memoria.
Il "registro di impaginazione", che fa parte dell'hardware controllato dal sistema operativo, indica qual'è la ROM attiva in un particolare istante e può accedere in qualsiasi momento a un massimo di 16 ROM impaginate. Queste vengono esaminate dal SO quando si verifica un evento non previsto. Se, ad esempio, il SO riceve un comando * che non riesce a identificare, interroga le diverse ROM impaginate, presenti nel computer, prima di emettere il messaggio d'errore BAD COMMAND. La ROM del SO, ad esempio, non riconosce il comando *INFO che viene quindi passato a un'altra delle ROM impaginate: in questo caso, il comando viene accettato ed eseguito dalla ROM DFS.

Versioni del BASIC BBC

Per i Micro BBC venduti in Gran Bretagna sono state sviluppate due versioni di BASIC: il BASIC I e il BASIC II. Per scoprire la versione residente, l'utente deve premere CTRL-Break, digitare il comando REPORT e battere il tasto Return. Se sullo schermo compare la scritta di copyright datata 1982, il Micro usa il BASIC II, se invece l'anno è il 1981, il dialetto residente è il BASIC I. La versione più recente presenta meno difetti ed un maggior numero di funzioni della precedente.
Le nuove funzioni comprendono il comando d'archiviazione OPENUP e una versione modificata dell'istruzione OPENIN. Mentre il comando OPENIN del BASIC II apre un file solo in ingresso, quello del BASIC I lo apre sia in ingresso che in uscita, dimostrandosi uno strumento prezioso per la gestione su disco dei file ad accesso casuale. Il comando OPENUP del BASIC II apre i file sia in ingresso sia in uscita (equivale quindi all'OPENIN del BASIC I). La modifica di OPENIN comporta dei problemi nello scambio dei programmi tra gli apparecchi. Se infatti si carica su una macchina che usa il BASIC I un programma scritto in BASIC II che contiene delle istruzioni OPENIN, il primo dialetto non riconosce il token usato dal secondo per rappresentare il comando OPENIN. Quindi la seguente istruzione in BASIC II:

```
Y% = OPENIN(COSTI)
```

viene letta dal BASIC I come:

```
Y% = (COSTI)
```

e causa la comparsa di un messaggio d'errore.
Le altre nuove funzioni del BASIC II interessano soprattutto i programmatori in codice macchina. Se, programmando in BASIC I, occorre inserire un dato in un programma di codice macchina, è necessario abbandonare l'assemblatore del BBC ed usare gli operatori ?, ! e $ per caricare il dato in memoria. Per esempio:

```
1000 LDA dato
1010 .dato
1020 ]
1030 ?P% = 00
1040 P% = P% + 1
1050 OPT pass
```

In alternativa a questa lunga procedura, molti assemblatori offrono la possibilità di adoperare "direttive" (ossia pseudoistruzioni), che ordinano all'assemblatore di eseguire un particolare lavoro: nel programma precedente, OPT è una direttiva dell'assemblatore. Il BASIC II possiede quattro direttive non previste dal BASIC I: EQUB riserva un byte di memoria, EQUD ne riserva due ed EQUW tre; EQUS, infine, permette di memorizzare una stringa. Quindi, in BASIC II, il programma precedente potrebbe essere scritto così:

```
1000 LDA dato
1010 .dato EQUB 00
1020 ... resto del programma
```

Questa volta nel registro A viene caricato il byte contenuto all'indirizzo dato che in questo caso è 0. Nell'esempio successivo:

```
1000 .messaggio EQUS "Salve !
```

byte che contengono il messaggio Salve ! vengono caricati in memoria all'indirizzo indicato dall'etichetta .messaggio. Tutte queste direttive dell'assemblatore aggiornano automaticamente P% in base all'informazione introdotta.
Un'altra interessante novità del BASIC II è il comando OSCLI. È già stato menzionato come sia possibile passare comandi * al sistema operativo del BBC chiamando OSCLI all'indirizzo &FFF7 (vedere a pag. 1377). In BASIC I è

necessario depositare nelle variabili X% e Y% (oppure caricare nei registri X e Y) l'indirizzo in memoria della stringa che rappresenta il comando da comunicare. Come dimostra la tabella, il nuovo comando OSCLI del BASIC II semplifica questa operazione:

BASIC I	BASIC II
$&A00 = "*TAPE" X% = &A00 MOD 256 Y% = &A00 DIV 256 CALL &FFF7	OSCLI("*TAPE")

Oltre a queste due versioni del BASIC, esistono una versione del BASIC II modificata per il mercato americano, che permette di gestire differenti caratteristiche di visualizzazione, ed una versione chiamata HI BASIC, dedicata al secondo processore 6502, che sembra essere una combinazione del BASIC II e della sua versione americana. Dal punto di vista della programmazione, le funzioni e gli obiettivi di quest'ultima versione sono identici a quelli del BASIC II. Comunque sia, i dialetti BASIC I e II usano la memoria del Micro BBC allo stesso modo.

Area di lavoro del linguaggio

Lo spazio di memoria compreso tra gli indirizzi &400 e &7FF viene chiamato "area di lavoro del linguaggio" ed è riservato al linguaggio correntemente in uso, come pure la zona di memoria in pagina 0, compresa tra &00 e &6F. Tuttavia alcune zone di quest'ultima, soprattutto quelle poste tra gi indirizzi &50 e &6F, vengono impiegate solo raramente. Il BASIC usa la pagina 4 della memoria per immagazzinare le variabili.
L'area compresa tra &400 e &46B accoglie i valori assegnati alle variabili intere residenti, come A% o X%; la variabile @% viene memorizzata nei quattro byte da &400 a &403 (il byte meno significativo del suo valore si trova all'indirizzo &400), A% da &404 a &407 e così via fino all'ultima variabile residente Z%.
L'area di memoria compresa tra &480 e &4F9 viene usata come un indice dall'interprete del BASIC e punta agli indirizzi di memoria contenenti le variabili intere o a virgola mobile, le matrici o le variabili di stringa. Ad esempio, i due byte agli indirizzi &482 e &483 puntano ad un'area di memoria, compresa tra TOP e HIMEM, contenente le informazioni che riguardano tutte le variabili in uso il cui nome inizia per A. L'indirizzo registrato nelle locazioni &4F6 e &4F7 punta invece alle procedure BASIC archiviate in memoria mentre quello in &4F8 e &4F9 si riferisce alle funzioni.
L'area di memoria compresa tra &500 e &5FF viene usata dall'interprete del BASIC come stack, che però non va confuso con quello del chip 6502, situato in pagina 1 della memoria. Il BASIC adopera la pagina 5 per immagazzinare le informazioni relative agli indirizzi di ritorno dalle chiamate GOSUB e i dati necessari per una corretta esecuzione dei cicli iterativi FOR...NEXT e REPEAT...UNTIL.

Il BASIC usa la pagina 6 della memoria per gestire le variabili stringa, quando viene adoperata la versione estesa del comando CALL, per eseguire un programma in codice macchina (questa viene esaminata più avanti nel corso). La pagina 7 svolge la funzione di buffer d'ingresso, dove vengono "parcheggiati", in attesa di elaborazione, i comandi BASIC o linee di programma immesse da tastiera: i comandi diretti vengono poi interpretati ed eseguiti, mentre le linee di programma vengono inserite in posizione corretta all'interno del programma. La nostra panoramica sull'uso dell'area di lavoro del Micro BBC da parte del BASIC termina qui. Si ricordi che, quando il BASIC non viene usato, quest'area di memoria resta disponibile.

Gestione degli errori in BASIC

Programmando in BASIC, spesso accade di commettere errori che vengono successivamente segnalati dal computer con adeguati messaggi. Un'interessante caratteristica del sistema di gestione degli errori nel BASIC BBC è la possibilità data all'utente di sfruttare la routine di trattamento degli errori per produrre propri comandi d'errore, completi di messaggi e di codici numerici. Questa opzione risulta molto utile quando si elaborano programmi in codice macchina, da usare abbinati al BASIC. La routine che tratta gli errori generati da un programma BASIC viene puntata dal valore contenuto in un vettore chiamato BRKV, con indirizzo &202 e &203. Vediamo adesso la procedura per lanciare la routine puntata da questo vettore.
Quando il BASIC incontra un errore di programmazione, ad esempio una divisione per zero, ne segnala la presenza eseguendo un comando di codice macchina chiamato BRK. Questo ordina al 6502 di passare il controllo alla routine il cui indirizzo è contenuto nel vettore BRKV. In condizioni normali, l'istruzione BRK per la routine è seguita da una serie di byte con questo formato:

BRK
Codice d'errore (un byte)
Messaggio d'errore (una serie di byte)
Fine del messaggio (un byte zero)

In presenza del codice d'errore 4, che genera il messaggio Mistake, viene eseguita la seguente routine per richiamare quella di trattamento dell'errore.

Byte	Descrizione
BRK	Istruzione
4	Codice d'errore
M	
i	
s	Codici ASCII
t	o messaggio
a	d'errore
k	
e	
00	Fine del messaggio

La routine di trattamento dell'errore del BASIC assegna alla variabile di sistema ERL il numero della linea che contiene l'errore e alla variabile ERR il relativo codice d'errore, in questo caso 4. Infine la routine controlla la versione del BASIC in uso (REPORT) e visualizza il messaggio Mistake.
Chi programma in codice macchina può usare l'istruzione BRK all'interno dei propri programmi e definire propri messaggi d'errore. Infatti l'istruzione ON ERROR GOTO del BASIC accetta qualsiasi messaggio d'errore definito dall'utente come se fosse un errore generato dall'interprete del BASIC. Per esempio, la seguente sezione di un programma in codice macchina visualizza il messaggio d'errore Numero troppo grande:

```
1010 BRK
1020 EQUB 254
1030 EQUS "Numero troppo grande"
1040 EQUB 00
```

Visualizzando il contenuto della variabile ERR si trova che esso è 254 e impartendo REPORT si ottiene il messaggio Numero troppo grande. Volendo, è possibile anche cambiare il contenuto del vettore e modificare la procedura seguita dal Micro BBC per rispondere agli errori. Inoltre questo vettore permette di aggiungere nuovi comandi al BASIC del BBC, sebbene questa operazione richieda lo sviluppo di programmi piuttosto sofisticati.
Nella prossima sezione vengono approfondite le interazioni tra il BASIC ed il sistema operativo del Micro BBC.

ANALISI DELLE ROUTINE

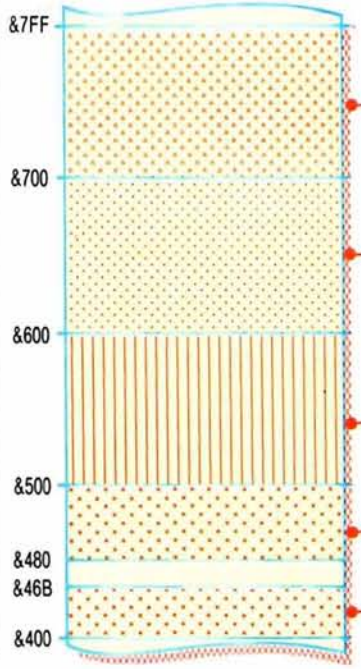
La tabella contiene gli indirizzi ROM di alcune delle routine BASIC del Micro BBC sin qui citate. Il loro studio permette di comprendere meglio il funzionamento del BASIC. Poiché non tutte le routine della tabella terminano con un'istruzione RTS, è consigliabile non usarle indiscriminatamente all'interno dei propri programmi. Tuttavia un'analisi approfondita delle tecniche impiegate può contribuire a migliorare le capacità di programmazione

Indirizzi d'ingresso in ROM &800

Istruzione BGET	&BF6F
Istruzione BPUT	&BF58
Istruzione CHAIN	&BF2A
Istruzione CLOSE	&BF99
Funzione EOF	&ACB8
Funzione INKEY	&ACAD
Funzione INKEY$	&B026
Istruzione INPUT	&BA44
Istruzione INPUT#	&B9CF
Comando LOAD	&BF24
Funzione OPENIN	&BF78
Funzione OPENOUT	&BF7C
Funzione OPENUP	&BF80
Istruzione PRINT#	&8D2B
Comando SAVE	&BEF3
Istruzione TIME	&92C9
Funzione TIME	&AEB4
Istruzione VDU	&942F

Per gentile concessione della Acorn Computers. © Acorn Computer Ltd (1982)

L'AREA DI LAVORO DEL LINGUAGGIO



Le locazioni di memoria comprese tra gli indirizzi &400 e &7FF sono riservate al linguaggio in uso. Lo schema mostra la suddivisione dell'area durante l'impiego del BASIC. La mappa di memoria completa del Micro BBC è a pag. 1359

Buffer d'ingresso del BASIC. Accoglie comandi diretti (eseguiti immediatamente) e linee di programma (che vengono trasferite da questa area all'interno del programma)

Area di lavoro per le variabili stringa e per il comando esteso CALL

Area dello stack del BASIC. usata per immagazzinare informazioni relative alle RETURN relative alle istruzioni GOSUB e per cicli FOR...NEXT e REPEAT...UNTIL

Indice per le altre variabili intere, a virgola mobile, di stringa e per le matrici

Area di immagazzinamento dei valori assegnati alle variabili intere residenti (@%, A%, B%, ... Z%)

LIZ DIXON

DIFFERENZE TRA LE VERSIONI BASIC I E II DEL BBC

Comando o Bug	BASIC I	BASIC II
REPORT	Messaggio della versione 1981	Messaggio della versione 1982
OPENUP	Assente	Apre un file in lettura e scrittura (simile al comando OPENIN del BASIC I)
OPENIN	Apre un file in lettura e scrittura	Apre un file solo in lettura
OPT n	Possiede solo i comandi OPT 1 - 3	Possiede i comandi OPT 1 - 7. I quattro nuovi comandi (OPT 4 - 7) sono simili ai primi tre, ma il codice viene assemblato all'indirizzo dato dalla variabile O%
O%	Nessun impiego particolare	Abbinato alla funzione precedente (comandi OPT n)
EQUB EQUD EQUS EQUW	Assenti	Consentono al programmatore di inserire dati nei programmi in codice macchina senza lasciare il programma assemblatore
INSTR (a$,b$)	Comando difettoso: se a$ è minore di b$, il programma può bloccarsi	Corretto nella versione BASIC II
Numeri Reali	Memorizzati con una precisione di nove cifre	Memorizzati con una precisione di dieci cifre
OSCLI	Assente	Consente di trasferire con facilità comandi dalle variabili stringa BASIC al SO

Elite

Un gioco basato su un'incredibile varietà di generi

L'*Elite* della Acornsoft è un cocktail delle caratteristiche tipiche dei giochi arcade, d'avventura e di strategia. L'obiettivo del gioco è quello di divenire membri dell'élite dei commercianti dello spazio. Per ottenere questo riconoscimento, il giocatore deve commerciare con successo su alcuni tra le migliaia di pianeti di otto galassie.
Per sopravvivere abbastanza a lungo e accumulare ricchezze sufficienti per entrare nella ristretta cerchia dell'élite, il giocatore deve trasformarsi non solo in esperto astronauta e baldo combattente, ma anche in spregiudicato affarista, sempre disposto a correre sul "filo" della legge. Infatti, il giocatore può accrescere rapidamente i propri guadagni facendo contrabbando tra i pianeti o diventando un pirata dello spazio. In tal caso, però, la polizia spaziale, dopo averlo schedato come delinquente ricercato, inizia a dargli la caccia con le proprie veloci astronavi.
All'inizio del gioco, la nave Cobra Mark III del giocatore (armata di un laser a impulsi frontali e con una capacità di carico di 20 tonnellate), si trova ancorata a una stazione spaziale del pianeta Lave. Prima di lanciarsi nello spazio, il giocatore deve imparare a usare i numerosi comandi e quadranti di controllo e addestrarsi nelle manovre di attracco, altrimenti rischia di fracassarsi al primo volo! Inoltre, se le manovre di avvicinamento alle stazioni orbitanti attorno ai pianeti ostili risultano troppo lente, l'astronave attrae l'attenzione dei pirati.
Ancorata la nave, iniziano gli scambi commerciali. Confrontando i prezzi di mercato del pianeta con le merci in suo possesso, il giocatore decide se gli conviene acquistare o vendere per realizzare il maggior profitto. Uno dei principali difetti dell'*Elite* (del resto comune alla maggior parte dei giochi "commerciali" prodotti per i computer), è l'impossibilità di contrattare i prezzi delle merci. Il giocatore deve prendere o lasciare!
Un buon affarista deve saper decidere rapidamente quali merci acquistare e quali vendere. Su un pianeta agricolo, ad esempio, conviene acquistare prodotti alimentari a basso costo da vendere a prezzi elevati su un pianeta industriale, per poi ripartire da qui con macchinari da smerciare su un pianeta agricolo. Il giocatore deve valutare anche l'organizzazione politica del pianeta e gli alieni che lo abitano. Un pianeta politicamente debole è spesso rifugio di pirati, mentre alcune razze aliene possono risultare più truffaldine o pericolose di altre.
L'*Elite* è dotato di un'ottima grafica tridimensionale. Inizialmente i pianeti, le stazioni spaziali e le astronavi, disegnati in forma schematica, sembrano complicare il gioco, ma in seguito permettono di apprezzare maggiormente l'alta risoluzione delle immagini e la rotazione tridimensionale degli oggetti. Questo tipo di grafica produce effetti stupendi nelle fasi di aggancio alle stazioni orbitali e di combattimento. Le stazioni possiedono un solo ingresso che deve essere imboccato con la giusta inclinazione. Inoltre, la possibilità di "vedere" la nave nemica (ne esistono svariati modelli), durante il combattimento, accresce notevolmente il realismo del gioco.
L'*Elite*, sia in formato cassetta che su disco, è corredato da un manuale che illustra i diversi aspetti del gioco, da un racconto intitolato *The Dark Wheel* che ne riprende la trama, da una lista dei comandi di controllo, da una mascherina per i tasti di funzione e da un poster che raffigura i diversi tipi di astronave in gioco. Non deve quindi sorprendere che l'*Elite* sia il gioco più venduto tra quelli prodotti per il Micro BBC.

Elite: Per il Micro BBC e l'Electron (il joystick è facoltativo)
Editori: Acornsoft Ltd, (GB)
Autori: Ian Bell, David Braben
Formato: Su disco o cassetta

Un gioco... per pochi!
La prima di queste tre immagini tratte dall'Elite", è quella d'apertura e mostra l'astronave Cobra MarK III. Il gioco può durare molto a lungo, ma è possibile interromperlo e salvarlo. Nella parte inferiore dello schermo sono rappresentati i vari quadranti di controllo del volo e una mappa radar tridimensionale dello spazio attraversato.
La seconda immagine mostra le stelle nelle vicinanze dell'astronave. Posizionando la griglia di puntamento su un particolare corpo celeste si ottengono informazioni sul tipo di governo, sull'economia ecc. del pianeta puntato. La terza figura rappresenta una stazione spaziale. Per attraccare il giocatore deve pilotare l'astronave all'interno del piccolo ingresso rettangolare

Inizia l'avventura





Diagramma a breve raggio



La stazione spaziale

Necessità particolari

Il computer può essere uno strumento insostituibile per le persone handicappate

Molte persone, adulti e bambini, possono avere speciali necessità nell'apprendimento, sia a causa di fattori fisici (sordità o cecità) sia per cause mentali. In certi casi alcune di queste necessità sono piuttosto ovvie, ma in altri è talvolta difficile prevederle o perlomeno individuarle, poiché sono specifiche di ciascuna invalidità. Va detto anche che tali difficoltà spesso provengono più dall'incomprensione della società che non dalla invalidità vera e propria.

I tentativi, compiuti da organismi pubblici, di sviluppare corsi specializzati basati su microcomputer sono molto rari, soprattutto per mancanza di attrezzature adeguate e, in seconda istanza, per carenza di personale specializzato.

Ne consegue che sono davvero poche le ditte produttrici di software e di hardware specifici presenti su un mercato di così limitate potenzialità. Esistono tuttavia gruppi di volontari che si dedicano con impegno al problema degli handicappati, anche sotto il profilo tecnologico.

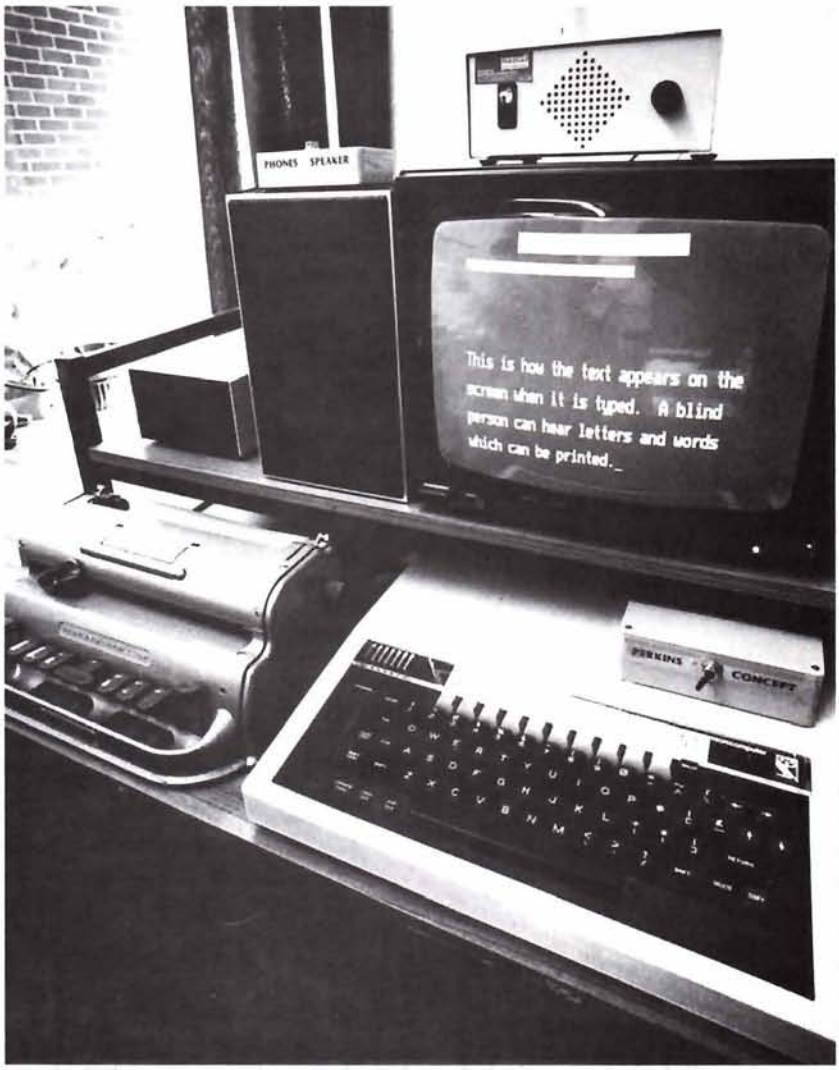
In Gran Bretagna sono sorti alcuni centri (gli Special Education Microelectronics Resources Centres, o SEMERC) che si dedicano in particolare alla formazione e informazione di insegnanti sulle più recenti acquisizioni tecnologiche. La mole di lavoro di questi centri è colossale (spesso un solo centro si occupa di oltre 700 scuole), nonostante le limitate risorse economiche.

La situazione diviene ancor più critica a causa degli innumerevoli problemi che di continuo sorgono in questa attività: una delle maggiori difficoltà, ad esempio, è quella di comunicare con le persone portatrici di handicap, anche se i più recenti sviluppi tecnologici hanno aperto nuovi orizzonti in tal senso.

Un apparecchio utilissimo, di recente fabbricazione, è il Photonic Wand (letteralmente "bacchetta fotonica"), costituito da un sensore ottico montato su di un casco: sfruttando i movimenti della testa è possibile (anche a persone prive di parola o che non possono usare gli arti) far funzionare un Micro BBC. Il dispositivo è collegato al computer tramite la porta analogica e il sensore posto sul casco funziona in modo simile alle normali penne ottiche, facendo muovere il cursore sullo schermo.

Grazie a questo semplice dispositivo, la persona può adoperare uno speciale word processor, chiamato *Write*, in cui le lettere maiuscole e minuscole, i vari numeri e segni e le stesse funzioni (memorizzazione, richiamo ecc.) vengono selezionati per mezzo del sensore.

Con lo stesso criterio, un altro programma,



TONY SLEEP

POSTO DI LAVORO

La "stazione di lavoro Vincent" racchiude in sé molti adattamenti hardware che ne permettono l'uso a persone colpite da handicap di vario genere. Una speciale stampante in Braille consente l'emissione di stampati comprensibili ai non vedenti, mentre un sintetizzatore vocale fornisce una forma di risposta udibile. Il testo sullo schermo è riprodotto in dimensioni maggiori rispetto al normale, per agevolare chi ha problemi di vista e, infine, al sistema è applicabile una "tastiera concettuale", che facilita l'immissione di comandi. Purtroppo il costo delle stampanti in Braille è ancora molto alto, limitandone la diffusione

Paint, consente di disegnare immagini sullo schermo e un altro ancora, *Music*, di comporre e suonare brani musicali.
Sono stati fatti, inoltre, alcuni tentativi di estendere l'uso della "bacchetta fotonica" a sistemi di sintetizzazione vocale, collegati all'apparecchio telefonico: la persona può selezionare sul computer alcune frasi "tipo" (ad esempio «può ripetere la frase, per favore»), che vengono emesse da un sintetizzatore vocale de D E Systems posto vicino al telefono. In un prossimo futuro dovrebbe essere disponibile un sistema più sofisticato, dotato di un vocabolario più ricco. Comunque, anche il dispositivo attuale rappresenta, sia pure nella sua semplicità, un enorme passo in avanti.
Per le persone colpite da cecità il problema è del tutto diverso, soprattutto perché i computer sono essenzialmente degli apparecchi progettati per persone vedenti: monitor, stampanti e plotter sono tutti dispositivi che implicano l'uso della vista.
Per risolvere questo problema, gli studi si sono rivolti a computer dotati di sintetizzatori vocali e interfacciati a speciali stampanti in Braille.
I programmi più sofisticati consentono anche la formattazione del testo, prima di inviarlo in stampa.
I margini, le testate e le spaziature vengono definite con tale precisione che il prodotto finale è indistinguibile da quello prodotto da una persona vedente.
Alcuni bambini handicappati non sono in grado di adoperare tastiere normali e quindi sono state progettate speciali tastiere a sfioramento, chiamate "tastiere concettuali", sulle quali l'insegnante sovrappone speciali mascherine plastiche, recanti simboli o scritte comprensibili al bambino, diversificando così l'utilizzo della tastiera a seconda delle circostanze specifiche.
Ad esempio, la mascherina per far muovere una tartaruga con il LOGO è suddivisa in quattro settori, su ognuno dei quali è riprodotta una direzione.
Esiste già una discreta biblioteca di programmi che funzionano in abbinamento alle "tastiere concettuali".
Il mouse, previsto da apparecchi quali il Macintosh ad esempio, aiuta indubbiamente a superare la "barriera" della tastiera ed ha grandi potenzialità applicative in questo campo.
Sempre in Gran Bretagna è sorto un gruppo di lavoro (AMASE - Apple Macintosh Applications in Special Education), che si dedica alle possibili applicazioni del Macintosh nell'insegnamento speciale. In alternativa al mouse, esistono speciali interruttori, di varie forme e dimensioni, da applicare alle parti del corpo sulle quali la persona handicappata può avere un certo controllo.



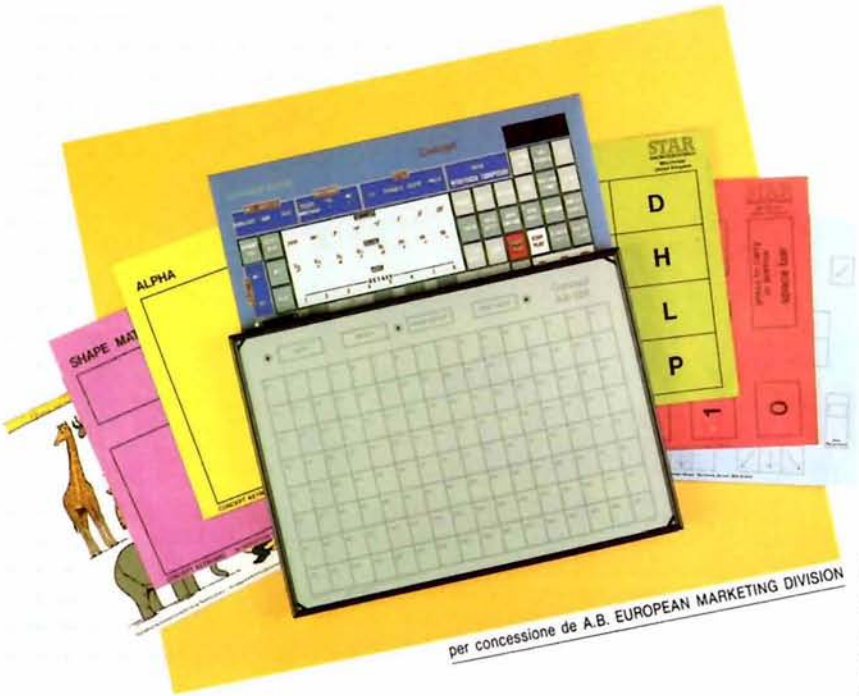


Tocco magico

Il "Photonic wand", o "bacchetta fotonica" è un dispositivo semplice ma utile per consentire la comunicazione, anche a persone portatrici di handicap, con il computer e, di conseguenza, anche a sintetizzatori vocali, word processor e tutta una serie di altre applicazioni. Un sensore ottico, montato su di un casco, converte i movimenti della testa in segnali e comandi per il computer

Molte "stazioni di lavoro" di questo genere sono state già sperimentate in scuole, abitazioni e posti di lavoro.
Per quanto riguarda il settore del software, sono già stati sviluppati molti word processor "parlanti", utilizzabili tramite la normale tastiera. Agendo sul tasto di cancellazione, addirittura, viene pronunciata la lettera eliminata. Il testo viene elaborato tramite un "cursore sonoro", che legge le parole e le frasi via via che scorre su di esse, facilitando così l'inserimento o la modifica di parti del testo memorizzato.

Uno di questi dispositivi viene applicato agli occhi e capta il movimento orizzontale o verticale di questi: il segnale viene poi amplificato e serve per comandare gli interruttori ad esso collegati.
In casi molto particolari, laddove è necessario focalizzare l'attenzione della persona handicappata che ha difficoltà di concentrazione vengono montate luci colorate e lampeggianti, ad esempio, sulla tartaruga da pavimento, in modo da poterne seguire il percorso.
In molte occasioni si è potuto notare che un simile espediente è un primo stimolo che fa reagire una persona apparentemente passiva. Alla conferenza annuale del British Logo User's Group, nel 1984, la dottoressa Sylvia Weir ha illustrato i notevoli progressi ottenuti, grazie all'impiego del LOGO e della tartaruga, da bambini handicappati fisicamente e mentalmente.
In particolare la dottoressa Weir ha citato il caso di un ragazzo diciassettenne ritenuto molto intelligente dai suoi insegnanti, che, incapace di scrivere una sola parola ha invece appreso subito i principi di programmazione del LOGO.
Due anni dopo egli ha iniziato a frequentare l'università e ha redatto un saggio: "Intelligenza in trappola". Un altro caso citato è stato quello di un bambino autista di 7 anni, che, eccitato per essere riuscito a impiegare la tartaruga, tramite una speciale scatola a pulsanti, ha parlato per la prima volta in vita sua.
Gli esempi potrebbero continuare all'infinito, in quanto di giorno in giorno si scoprono nuove possibilità applicative per aiutare chi, fino a pochi anni fa, non poteva disporre di canali didattici o espressivi adeguati.
Come l'istruzione tradizionale, anche quella orientata a persone portatrici di handicap si sta gradualmente arricchendo grazie ai notevoli sviluppi della tecnologia; tuttavia l'ostacolo maggiore, le cui conseguenze si fanno sentire in modo ancor più pesante che non sull'istruzione tradizionale, è la scarsità di fondi e il poco interesse, da parte del mondo industriale, a produrre apparecchiature specifiche dal costo accettabile. Nonostante ciò, grazie all'impegno di molti volontari, lo sviluppo sia del software che dell'hardware per queste particolari applicazioni continua a crescere.





Le tastiere concettuali
Su queste speciali tastiere "a sfioramento" è possibile sovrapporre mascherine sulle quali sono riprodotte immagini concernenti particolari argomenti. Ciò consente di scegliere l'immagine appropriata, evitando l'ostacolo costituito dalle tastiere di tipo tradizionale

I COMPUTER E LE DOTI INDIVIDUALI

Se è vero che il computer può favorire lo sviluppo mentale di bambini normali, ciò è ancor più vero nel caso di quelli handicappati. Spesso, infatti, bambini handicappati fisicamente o con difficoltà di percezione risultano notevolmente dotati per la loro età e ciò può essere evidenziato ricorrendo all'uso di computer, coi quali i bambini riescono facilmente a prendere confidenza.
In generale, ecco cosa può offrire un computer ad un bambino:

- L'opportunità di sviluppare le proprie doti e interessi ad un ritmo del tutto adeguato alle proprie capacità
- Criteri espressivi totalmente nuovi, anche nell'affrontare problemi
- L'opportunità di svolgere esercizi, altrimenti noiosi e ripetitivi, in un contesto stimolante che mantiene vivo l'interesse
- Un nuovo strumento (e un nuovo campo) da esplorare

Un programmatore di EPROM

Conservare i propri programmi su EPROM è comodissimo

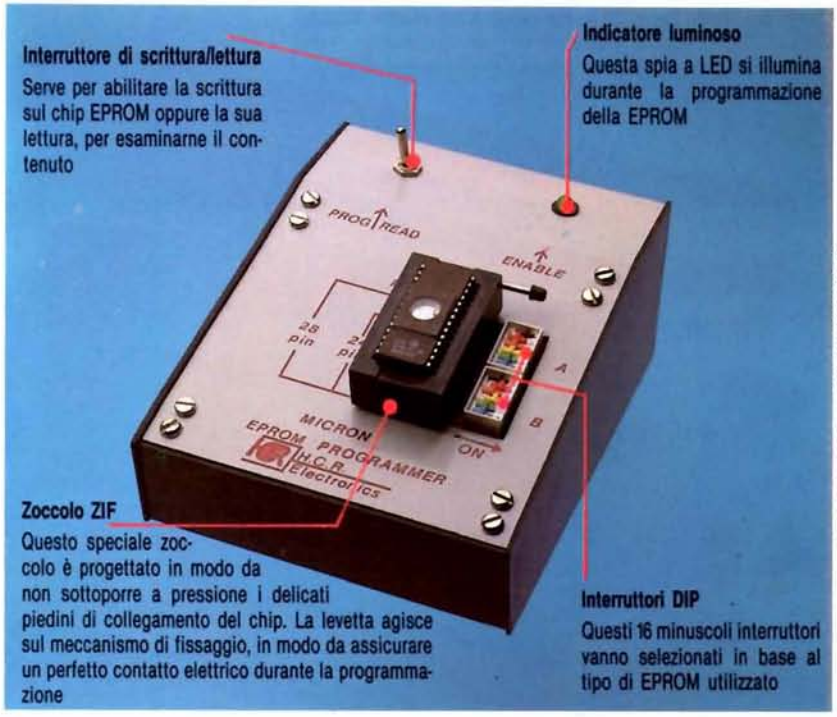
Componenti esterne

Un programmatore di EPROM consente di conservare in forma permanente i propri programmi, che vengono prima trasferiti in memoria per poi essere "fissati" nei circuiti elettrici del chip di EPROM



Esame ravvicinato

Il chip della EPROM è facilmente visibile attraverso la finestrella sull'involucro che lo protegge. Con una lente d'ingrandimento è possibile intravedere la matrice dei contatti che formano la memoria



CHRIS STEVENS - PER CONCESSIONE DELLA TEXAS INSTRUMENTS

Una delle maggiori doti del Micro BBC è la sua flessibilità d'impiego. Nel progettare l'apparecchio, la Acorn ha pensato bene di includere quattro zoccoli per ROM, che l'utente può utilizzare per inserire programmi applicativi. Il passaggio da una ROM all'altra avviene con semplici comandi "stella". Il vantaggio di conservare programmi su ROM è indiscutibile: il loro caricamento risulta più veloce sia rispetto al nastro che ai dischetti e, inoltre, non c'è il rischio di cancellarli o danneggiarli involontariamente.

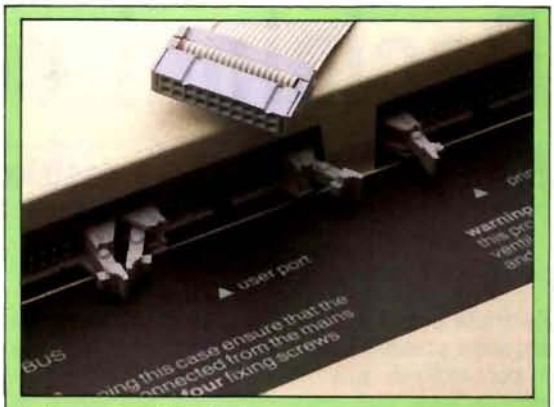
In commercio esiste una vasta gamma di programmi su ROM "già pronti": word processor, database, linguaggi e altri ancora. Per i molti possessori di BBC che desiderano inserire i propri programmi su ROM, esiste un apposito apparecchio, chiamato appunto "programmatore di EPROM".

Come tutte le memorie di tipo ROM (ossia a sola lettura), le EPROM consistono in una matrice di linee elettriche suddivise in righe e colonne: ogni intersezione corrisponde a una cella di memoria. Se all'intersezione è applicata una tensione elettrica, essa viene considerata a livello logico 1, altrimenti a 0. Il chip è dotato di un circuito di indirizzamento (otto linee, ossia un byte) grazie al quale è possibile selezionare un singolo gruppo di otto intersezioni, che costituiscono un byte di memoria. A questo punto le tensioni eventualmente presenti nelle intersezioni vengono convogliate sulle otto linee di uscita dei dati e passate alla CPU del computer.

In genere, quando un chip di EPROM esce dalla fabbrica, tutti gli stati logici della matrice sono posti a 1, per cui il contenuto di ogni byte di memoria vale &FF; facendo passare un'elevata corrente attraverso una combinazione di intersezioni, il collegamento viene "bruciato", portandone il livello logico a zero. Per programmare una EPROM, dunque, occorre per prima cosa preparare la configurazione di bit sulle linee di indirizzamento, per individuare un particolare byte di memoria, e poi inviare una tensione adeguata lungo le linee dei dati. In effetti, su ogni linea dei dati viene applicato un impulso di 25 volt per una durata di 50 microsecondi, quanto basta per "bruciare" il collegamento e programmare il particolare byte.

Per conservare i codici binari di un intero programma basta ripetere l'operazione, avendo cura di incrementare l'indirizzo di 1 ad ogni nuovo passo. Tuttavia, contrariamente a quanto accade sulle ROM vere e proprie,

in cui il processo di programmazione è irreversibile, le EPROM si possono riutilizzare se il programma non serve più.
A questo scopo sulla parte superiore dell'involucro del chip è presente una piccola finestrella di quarzo, che consente il passaggio della luce. Colpendo la finestrella con una dose adeguata di raggi ultravioletti, lo stato elettrico dei vari contatti viene riportato alle condizioni originarie cancellando, di fatto, tutte le configurazioni preesistenti. Ciò consente di riprogrammare il chip.
Il programmatore di EPROM della HCR Electronics consente di immettere fino a 16 Kbyte di programmi o dati su di un chip di EPROM. L'apparecchio è racchiuso in una piccola scatola, dalla quale sporge lo speciale zoccolo ZIF (con forza d'inserimento zero). Questo zoccolo consente l'inserimento e l'estrazione dei chip senza correre il rischio di danneggiarne i delicati piedini di collegamento. Lo ZIF è progettato in modo da accogliere sia i chip a 24 piedini, sia quelli a 28 piedini. Accanto allo zoccolo ci sono due serie di interruttori DIP (Dual In line Package), che permettono di selezionare il funzionamento del programmatore di EPROM. Un altro interruttore serve per abilitare la lettura o la scrittura del chip. È ovvio che la posizione di tale interruttore va sempre controllata, prima di dare tensione all'apparecchio, per evitare di cancellare inavvertitamente il contenuto di una EPROM già programmata.
L'apparecchio viene collegato, tramite un apposito cavetto, alla porta utente del Micro BBC. A questo punto, adoperando il software di corredo, si possono eseguire LOAD e SAVE di interi blocchi di byte, passando attraverso una zona riservata di memoria. Il programma è su cassetta ed è consigliabile, per prima cosa, trasferirlo in EPROM.
Il programma parte visualizzando un menu con tutte le opzioni offerte. Per copiare un programma in EPROM questo deve essere prima caricato in una zona di memoria (accantonata automaticamente dal software a partire dall'indirizzo &2000), usando l'opzione (L)oad. Fatto ciò si può passare all'opzione (P)rogram: sullo schermo compare un elenco dei vari tipi di EPROM programmabili, dal quale occorre selezionare quello desiderato. A seguito della scelta, sullo schermo compare la disposizione da dare ai 16 interruttori DIP. Gli interruttori la cui posizione è variata rispetto alla selezione precedente sono evidenziati facendo lampeggiare la relativa scritta. Dopo essersi accertati della corretta posizione dell'interruttore scrittura/lettura, basta premere un qualsiasi tasto per avviare l'operazione, durante la quale viene visualizzato l'indirizzo della cella in programmazione.
Il tempo occorrente per l'intero processo dipende in gran parte dal numero di celle il cui valore rimane quello originale, ossia &FF. Mediamente il tempo necessario va dai 74 secondi fino a 14 minuti per una EPROM da 16 Kbyte. Usando il comando (V)erify, è possibile controllare se la programmazione della EPROM è avvenuta in modo corretto. Se non vengono riscontrati errori, la EPROM è pronta per essere inserita in uno zoccolo del BBC.
Il comando G permette di inserire una breve routine prima del programma vero e proprio, in modo che il chip risponda a particolari comandi stella. Esistono diverse possibilità di registrazione a seconda che si tratti di programmi in BASIC o in codice macchina.
Tirando le somme, questo apparecchio risulta facile da usare ed è un ottimo strumento per migliorare il funzionamento del computer: poter produrre in proprio EPROM con i programmi preferiti o di uso più frequente è un vantaggio da non sottovalutare, soprattutto a chi intenda commercializzare i propri programmi in un formato molto efficiente.

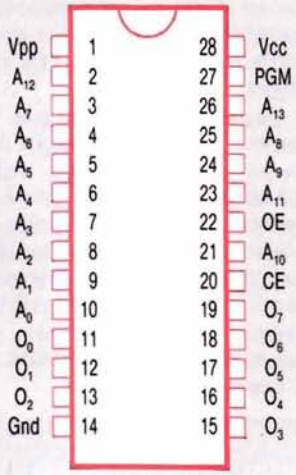


Collegamento dell'interfaccia
Il programmatore di EPROM è collegato alla porta utente del Micro BBC tramite un cavetto a 12 vie

DISPOSIZIONE DEI PIEDINI

Sigla	descrizione
A_0—A_{13}	Indirizzi
CE	Abilitazione del chip
OE	Abilitazione in uscita
O_0—O_7	Uscita dei dati
PGM	Programmazione
Vpp Vcc	Alimentazione
Gnd	Massa

Questa è la rappresentazione schematica della EPROM TMS27128 da 16 Kbyte, una tra quelle programmabili dall'apparecchio esaminato. Il chip è dotato di 28 piedini, ma l'apparecchio accetta anche quelli con 24 piedini. I piedini di indirizzamento sono 14, mentre quelli per le linee in uscita sono 8



LIZ DIXON

PROGRAMMATORE EPROM MICRON

DIMENSIONI

mm. 140 x 105 x 65

INTERFACCE

Cavo a nastro con connettore da applicare alla porta utente del Micro BBC

DOCUMENTAZIONE

Tre paginette spiegano ciascuno dei nove comandi disponibili. Mentre l'apparecchio è facilissimo da usare, la documentazione è un po' troppo tecnica per un principiante

PRO

L'uso dell'apparecchio è estremamente facile: basta selezionare le opzioni desiderate da un menu. Anche il collegamento non presenta problemi

CONTRO

Non vengono fornite spiegazioni sulla possibilità di ampliare le prestazioni dell'apparecchio, né su come utilizzarlo con software diverso da quello fornito. La registrazione su EPROM di programmi in codice macchina è limitata dal fatto che essi non sempre possono essere rilocati per partire dall'indirizzo &2000, necessario al funzionamento del programma

Decisioni costruttive

Esaminiamo due strutture decisionali in PASCAL: la IF e la CASE

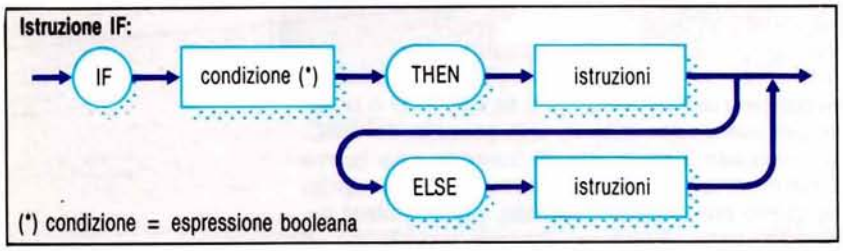
La frase condizionale IF del PASCAL è simile a quella presente anche in altri linguaggi, ma è certamente più flessibile, poiché l'intera espressione può estendersi su più linee di listato.
Per maggior chiarezza, conviene tabulare i vari elementi della frase, come nei seguenti esempi:

```
IF conto = limite
   THEN
      WriteLn ('Spazio esaurito')
   ELSE
      write ('Prossimo numero?');
IF numero > massimo THEN
   massimo: = numero
```

Nel secondo caso, l'assenza della parte ELSE equivale a:

```
ELSE
   {non far niente}
```

L'allineamento delle parole riservate THEN e ELSE (se presente) aiuta a individuare meglio il percorso attraverso i blocchi che compongono la struttura.
Il punto e virgola, che serve per terminare uno "statement", non va dunque mai posto prima della parola ELSE. Infatti, vale la pena di ricordare che l'istruzione vera e propria è la IF: mentre invece THEN e ELSE sono appendici che delimitano la condizione booleana e le azioni da intraprendere in conseguenza. Per esser più precisi, IF verifica sempre una condizione booleana: se questa è vera, viene eseguita l'espressione che segue THEN, altrimenti quella che segue ELSE.



Il PASCAL è stato il primo linguaggio a introdurre dati di tipo concettuale, noti con l'espressione "scalari enumerati". Questo nuovo genere di dati facilita una visione ad alto livello della classificazion dei dati, senza dover continuamente convertire, mentalmente, i dati in codici numerici (facili da ricordare per il computer, ma non per noi!). Abbiamo già visto come definire una semplice costante, ad esempio:

```
CONST
   larghezza = 80;
```

Una volta definito, il nome della costante larghezza può essere tranquillamente adoperato nel programma, al posto del numero che, ad esempio, indica il numero di colonne sullo schermo o sulla stampante. Per adattare il programma a uno schermo con 40 colonne, basta semplicemente modificare la sola linea che contiene tale definizione.
Supponiamo però di voler usare, in un programma, la grafica a colori: abbinare a ciascun colore un numero progressivo (ad esempio: rosso = 1, verde = 2, ecc.) è molto comodo per accrescere la leggibilità del listato. Tuttavia, se ci limitassimo a definire tali valori come costanti numeriche, potrebbe accadere che, più o meno consapevolmente, si tenti di estrarre la radice quadrata di un colore o di moltiplicare un colore per un altro! Oltre a essere piuttosto illogica, questa possibilità è una potenziale fonte di errori. Ecco dunque dove entrano in gioco gli scalari enumerati che, in sostanza, costituiscono un elenco di identificatori di costanti, delimitato entro una precisa gamma di valori. La definizione di questo tipo di dato concettuale, viene inserita nella sezione TYPE di un programma PASCAL. Ecco un esempio:

```
TYPE
   gmc = (rosso, verde, giallo, blu, magenta);
```

Poiché si tratta di una "definizione" (e non di una "dichiarazione"), la sintassi prevede l'uso del segno uguale, per associare all'identificatore del tipo (gmc) la serie di valori indicati tra parentesi.
Come sempre, in PASCAL, gli identificatori che formano la lista (nel nostro caso i colori) sono separati da virgole. Il valore di ciascuno di essi è automaticamente assegnato dal compilatore, partendo dallo zero. Per esaminare il valore attribuito a ciascun identifinuovo tipo di scalare, possiamo adesso dichiarare, cosicché ord(rosso) vale 0 e ord(verde) vale 1 e così via. Avendo definito questo una variabile nel modo consueto:

```
VAR
   colore : gmc;
```

In questo modo viene dichiarato un identificatore (colore) del tipo gmc, analogamente a quanto accade con:

```
VAR
   lettera : char;
```

dove si dichiara l'appartenenza di lettera alla classificazione caratteri.
Le sole operazioni consentite, sugli scalari enumerati, sono le relazioni di confronto e le funzioni scalari. Per esempio, possiamo scrivere:

```
if colore < giallo then
   colore: = succ(colore)
```

Si faccia attenzione a non usare, come magari verrebbe spontaneo, pred(rosso) o succ(giallo), che non esistono! La variabile colore è incompatibile con le variabili di altro tipo, scalare o no, quindi con questo sistema non è più possibile estrarre radici quadrate o usare espressioni del genere:

```
colore: = colore + 1
```

Ciò implica un'ovvia limitazione: i caratteri e i numeri possono essere adoperati come parametri per le write e WriteLn, ma è un errore scrivere

```
WriteLn(colore)
```

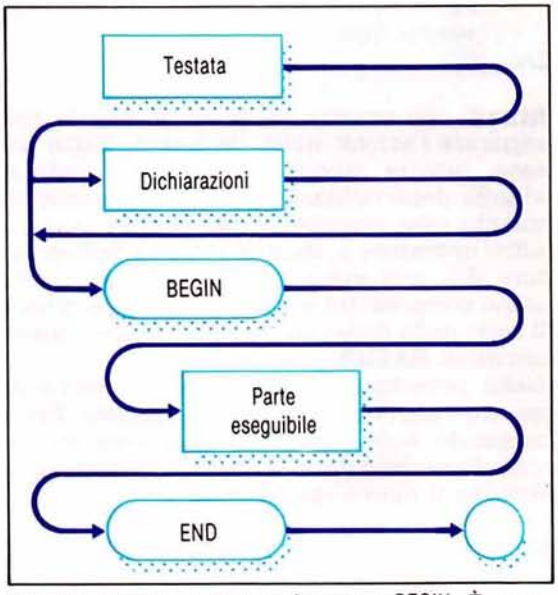
poiché i valori assegnati all'identificatore sono puramente concettuali: se vogliamo visualizzare i nomi che formano la lista occorre costruire una tabella di stringhe a parte.
Questo caso si adatta ad un'altra struttura prevista dal PASCAL, quella chiamata CASE. Il familiare formato della IF, talvolta, può complicarsi al punto da rendere difficoltosa la sua interpretazione. Si prenda ad esempio:

```
IF N=1
   THEN
      write('mo')
   ELSE
      IF N=2
         THEN
            write('do')
         ELSE
            IF N=3
               THEN
                  write('zo')
               ELSE
                  write('to')
```

In casi come questo, in cui la scelta del percorso dipende da un semplice scalare, interno ad una gamma limitata, è molto più conveniente usare l'istruzione CASE. Riferendosi al precedente esempio:

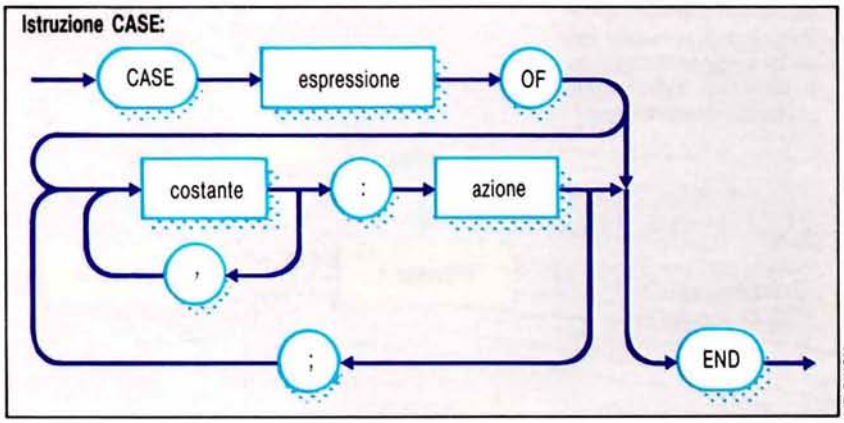
```
CASE N OF
     1          :write('mo');
     2          :write('do');
     3          :write('zo');
     4,5,6      :write('to');
END {CASE}
```

Si noti che questa struttura prevede solo i casi specificati dalle etichette contenute al suo interno e appartenenti ad una precisa gamma (da 1 a 6, nell'esempio), per cui tutti i valori che la variabile N può assumere nel corso dell'esecuzione devono essere elencati nella CASE, altrimenti i valori non previsti generano un errore.
Tuttavia, alcuni compilatori di PASCAL prevedono l'impiego di una parola riservata (OTHERWISE oppure OTHERS), utile per definire la diramazione da prendere qualora il valore della variabile non rientri nella gamma specificata. Si consulti il manuale del proprio PASCAL, per la corretta sintassi da usare.
Questo è l'unico caso, in PASCAL, in cui la parola END delimita una struttura senza che questa venga aperta da una BEGIN. È una pratica consolidata, oltre che una buona norma di programmazione, terminare ogni struttura CASE con una END descrittiva, come nel caso illustrato.



Scheletro di programma
Questo è un diagramma semplificato della sintassi di un programma in PASCAL, in cui vengono evidenziati soltanto gli elementi principali. Si noti che le parole riservate appaiono scritte in maiuscolo: ciò serve a sottolineare la loro qualità, appunto, di parole riservate rispetto alle altre. In quasi tutte le versioni di PASCAL, comunque, anche le parole riservate possono essere scritte in caratteri minuscoli

Il funzionamento di CASE è molto semplice: l'espressione scalare racchiusa tra CASE e OF viene valutata e confrontata, in successione, con tutte le costanti che sono riportate nella lista.
Quando viene trovata una corrispondenza, l'esecuzione esegue l'azione associata e solo quella. In altre parole, il programma ignora tutte le altre diramazioni che non corrispondono. Ovviamente, l'azione può non essere limitata ad un'unica funzione: si può anche specificare una procedura, purché inizi con una BEGIN e termini regolarmente con una END.
Può capitare che a certi valori della gamma non debba corrispondere alcuna azione: in



tal caso basta associare al valore un'espressione 'nulla', comunicando così al PASCAL di "non fare niente".
Ecco un esempio:

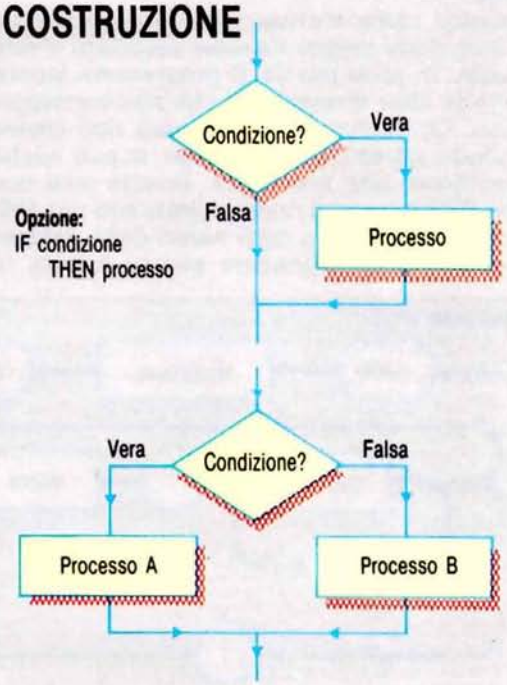
```
CASE N MOD 4 OF
    0 :non far niente;
    1,3 : begin
          write (N MOD 4 : 1,'quart');
          if N MOD 4 > 1 then
             write ('i')
          end;
    2 :write ('un mezzo')
END {CASE}
```

Si noti che occorre un punto e virgola per separare l'azione 'nulla' da quanto segue ad essa, mentre invece non occorre punto e virgola dopo l'ultima etichetta della lista, in quanto essa precede la parola END e non un altro operatore o etichetta. L'uso dell'operatore MOD, nell'esempio, assicura che i valori siano compresi tra 0 e 3. MOD, infatti, produce il resto della divisione fra interi (come accade anche in BASIC).
Nella prossima lezione vengono esaminati questo e altri operatori e varie funzioni. Per il momento ecco come CASE può risolvere la visualizzazione dei nomi dei colori, dopo aver definito il nuovo tipo di dato gmc:

```
CASE colore OF
    rosso       : write ('Rosso');
    verde       : write ('Verde');
    giallo      : write ('Giallo');
    blu         : write ('Blu');
    magenta     : write ('Magenta');
END {CASE}
```

SCEGLIERE UNA COSTRUZIONE

La costruzione IF...THEN esegue il "processo" soltanto se la condizione esaminata è vera. Se invece questa è falsa, l'esecuzione ignora il "processo" e passa all'istruzione successiva.
Mentre la costruzione IF...THEN offre un'opzione di percorso, la struttura IF...THEN...ELSE presenta invece una vera e propria scelta: in base al risultato della verifica eseguita al principio della costruzione viene eseguito uno tra due diversi "processi"



UN SEMPLICE PROGRAMMA

Il programma qui listato, che calcola semplicemente i giorni intercorsi fra la data immessa dall'operatore e l'ultimo giorno del mese, contiene numerosi richiami ai principi della programmazione in PASCAL.
L'istruzione CASE è usata per abbinare il dato numerico immesso a un dato di tipo "enumerato", tipico del PASCAL. Successivamente viene ricavato il valore abbinato per poter visualizzare una delle varie stringhe. Si noti l'impiego di una variabile char "di comodo" (simbolo), per tener conto di qualsiasi carattere venga usato dall'operatore come separatore tra i numeri della data.

```
PROGRAM          Date        (input, output);
CONST
    Punto = '.';
TYPE
    Calendario = (Gen, Feb, Mar, Apr, Mag, Giu,
                  Lug, Ago, Set, Ott, Nov, Dic);
VAR
    NomeMese         : Calendario;
    giorno,
    mese,
    anno,
    Rimanenti        : integer;
    simbolo          : char;
    Bisestile        : boolean;
BEGIN
  WriteLn ('Immettere la data nel formato :');
  WriteLn ('GG/MM/AA' : 40);
  WriteLn;
  write ('Data ? ');
  read (giorno, simbolo, mese, simbolo, anno);
  Bisestile := anno MOD 4 = 0;
                  { secoli non vengono considerati }
  IF (mese > 0) AND (mese < = 12)
    THEN
      CASE mese OF
        1       : NomeMese := Gen;
        2       : NomeMese := Feb;
        3       : NomeMese := Mar;
        4       : NomeMese := Apr;
        5       : NomeMese := Mag;
        6       : NomeMese := Giu;
        7       : NomeMese := Lug;
        8       : NomeMese := Ago;
        9       : NomeMese := Set;
        10      : NomeMese := Ott;
        11      : NomeMese := Nov;
        12      : NomeMese := Dic
      END       { CASE }
    ELSE
      BEGIN
        WriteLn  ('Eh ?');
        WriteLn  (' - il programma sta', ' per inchiodarsi !')
      END;      { NomeMese non è inizializzata! }
    CASE NomeMese OF
    Gen, Mar,
    Mag, Lug    : Rimanenti := 31 - giorno;
    Apr, Giu,
    Set, Nov    : Rimanenti := 30 - giorno;
    Feb         : IF Bisestile
                    THEN
                      Rimanenti := 29 - giorno
                    ELSE
                      Rimanenti := 28 - giorno
  END; { CASE }
  WriteLn;
  write ('Rimangono ', Rimanenti : 1, 'giorni in ');
  CASE NomeMese OF
        Gen     : write ('Gennaio');
        Feb     : write ('Febbraio');
        Mar     : write ('Marzo');
        Apr     : write ('Aprile');
        Mag     : write ('Maggio');
        Giu     : write ('Giugno');
        Lug     : write ('Luglio');
        Ago     : write ('Agosto');
        Set     : write ('Settembre');
        Ott     : write ('Ottobre');
        Nov     : write ('Novembre');
        Dic     : write ('Dicembre');
  END; { CASE }
  WriteLn (Punto)
END .
```

DATABASE

Ecco la decima tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi

DESCRIZIONE	INDIRIZZO
Remove FAC#1 from stack rimuove FAC#1 dallo stack	BD7E
Remove a string from stack rimuove una stringa dallo stack	BDDC
SAVE command routine routine comando SAVE	BEF3
SGN function routine routine funzione SGN	AB88
SIN function routine routine funzione SIN	A998
SOUND statement routine routine istruzione SOUND	B44C
SPC routine routine funzione SPC	8E58
SQR function routine routine funzione SQR	A7B4
STOP statement routine routine istruzione STOP	8AD0
STR$ function routine routine funzione STR$	B094
STRING$ function routine routine funzione STRING$	B0C2
Search for FN/PROC in catalogue cerca FN/PROC nel catalogo	945B
Search for a program line cerca una linea di programma	B99A
Search for line in program cerca una linea nel programma	9970
Search for name in catalogue cerca un nome nel catalogo	9469
Search for next DATA item cerca il successivo elemento DATA	BB50
Search program for a PROC/FN cerca FN/PROC in un programma	B112
Series evaluator calcolo di serie	A897
Set (&2A)=(&2A)*(&3F) pone (&2A)=(&2A)*(&3F)	9236
Set (&4B)=&46C pone (&4B)=&46C	A7F5
Set (&4B)=&471 pone (&4B)=&471	A7E9
Set (&4B)=&476 pone (&4B)=&476	A7ED
Set FAC#1=(&4B)+FAC#1 pone FAC#1=(&4B)+FAC#1	A500
Set FAC#1=(&4B)—FAC#1 pone FAC#1=(&4B)—FAC#1	A4FD
Set FAC#1=(&4B)/FAC#1 pone FAC#1=(&4B)/FAC#1	A6AD
Set FAC#1=0 pone FAC#1=0	A686

INDIRIZZO	DESCRIZIONE
B402	**BRK handling routine** routine per la gestione di BRK
B402	**Error handler** routine trattamento errori
B433	**BASIC error handler** routine trattamento di errori BASIC
B44C	**SOUND statement routine** routine istruzione SOUND
B472	**ENVELOPE statement routine** routine istruzione ENVELOPE
B4AO	**WIDTH statement routine** routine istruzione WIDTH
B4B1	**Assign value to numeric variable** routine per assegnare valori numerici alle variabili
B50E	**Print a character** visualizza un carattere
B545	**Print A in hexadecimal** visualizza A in esadecimale
B58A	**LISTO command routine** routine comando LISTO
B59C	**LIST command routine** routine comando LIST
B695	**NEXT statement routine** routine istruzione NEXT
B7C4	**FOR statement routine** routine istruzione FOR
B888	**GOSUB statement routine** routine istruzione GOSUB
B8B6	**RETURN statement routine** routine istruzione RETURN
B8CC	**GOTO statement routine** routine istruzione GOTO
B8E4	**ON ERROR OFF statement routine** routine istruzione ON ERROR OFF
B8F2	**ON ERROR statement routine** routine istruzione ON ERROR
B915	**ON statement routine** routine istruzione ON
B99A	**Search for a program line** cerca una linea di programma
B9CF	**INPUT# statement routine** routine istruzione INPUT#
BA44	**INPUT statement routine** routine istruzione INPUT
BAE6	**RESTORE statement routine** routine istruzione RESTORE
BB1F	**READ statement routine** routine istruzione READ
BB50	**Search forn next DATA item** cerca il successivo elemento DATA
BBB1	**UNTIL statement routine** routine istruzione UNTIL

Il Mouse della AMX

Un utile accessorio per il Micro BBC, corredato da alcuni programmi applicativi



L'editor per le icone consente di creare o modificare facilmente un intero "set" di icone



Il menu principale: il cursore è usato per selezionare uno tra i quattro programmi applicativi



Creazione di un disegno: prima si traccia il contorno



Le caratteristiche delle linee tracciate vengono selezionate dall'apposito menu

Il Mouse AMX

Questo utile accessorio hardware è fornito in un package completo, che include un chip di ROM con il software operativo, una serie di programmi applicativi su cassetta e su dischetto e due manuali molto esaurienti. Durante l'uso, il programma di disegno ("AMX Art program") rassomiglia molto al package grafico impiegato sul Macintosh della Apple, tuttavia il suo prezzo è notevolmente inferiore

Il package del mouse AMX, realizzato dalla Advanced Memory Systems (GB) per il Micro BBC, è veramente completo: comprende, oltre al mouse, il cavo necessario al collegamento con il computer (tramite la porta utente), un chip di ROM, due manuali e, infine, software su cassetta e su dischetto. Pur fornendo il software su cassetta, la AMS non nasconde il fatto che il package sia stato in effetti concepito pensando a chi possiede unità a dischetti.

L'apparecchietto, fabbricato in Giappone, è racchiuso in un involucro di plastica nera dal cui fondo sporge la sfera metallica necessaria per trasmettere il movimento. Il mouse è molto maneggevole e i tre pulsanti di comando (Movimento, Esecuzione e Cancellazione), collocati sulla faccia anteriore, sono facilmente raggiungibili dalle tre dita libere. Purtroppo, essendo di metallo, la sfera tende a scivolare su superfici troppo lisce (piani di formica o di legno lucidato), dove si rende necessario frapporre un foglio di carta per migliorarne il funzionamento.

All'interno, la sfera trasmette i movimenti a due cilindretti gommati, a ciascuno dei quali è fissato un disco. Questo è opportunamente forato e interposto tra una sorgente di luce e una cellula fotosensibile cosicché, ruotando, il fascio di luce viene di volta in volta o interrotto o lasciato passare attraverso i fori. Gli impulsi elettrici ottenuti sui capi della cellula fotosensibile vengono poi inviati al computer, che può così rilevare istante per istante l'esatta posizione e velocità del mouse.

Il software su ROM, necessario alla conversione e interpretazione dei segnali, provvede a comunicarli al computer. Le routine consentono l'impiego del mouse sia con software commerciale, sia con programmi di realizzazione propri. A tal riguardo, sono previste alcune routine molto utili: una fornisce le coordinate del mouse rispetto allo schermo del computer, una individua la pressione dei tasti di comando, una anima le icone raffigurate sullo schermo ed altre ancora. Sfruttando tali routine (alle quali si accede con un gruppo di comandi * di facile impiego) si possono realizzare programmi BASIC in cui l'uso del mouse si integra perfettamente.

Come accennato, la ROM contiene routine già predisposte per il software commerciale, grazie alle quali i segnali emessi dal mouse vengono interpretati come comandi impartiti sui normali tasti del cursore, mentre la funzione dei tre pulsanti di controllo può essere abbinata a qualsiasi tasto della tastiera (com-



IAN McKINNELL

IAN McKINNELL



Ecco che vengono aggiunti e "colorati" altri particolari del disegno



Le superfici interne possono essere distinte tramite diverse configurazioni di tratteggiature



Aggiungere scritte è molto facile: basta selezionare l'apposito menu, che si "srotola" su parte dell'immagine



I ritocchi finali si possono dare usando il funzionamento "a mano libera"

presi SHIFT e CTRL) a seconda delle esigenze. Il mouse può essere adoperato, ad esempio, per cancellare, muovere o copiare facilmente sezioni di testo elaborate con il word processor *View* della Acornsoft.

Tuttavia, le doti migliori del mouse si apprezzano usando programmi applicativi scritti appositamente. Il package AMX comprende due di tali programmi, che dimostrano con molta efficacia la duttilità del mouse e i vantaggi che derivano dal suo impiego. Il primo è un programma BASIC che serve per disegnare icone da immettere nei propri programmi. Le icone non sono che particolari simboli o immagini grafiche visualizzate dai programmi per facilitare, ad esempio, la scelta da un menu o per simboleggiare il cursore. Il programma generatore di icone fornisce una griglia di 16 x 16 quadrati, nella quale costruire l'immagine voluta: basta "accendere" o "spegnere" opportunamente ciascun quadratino. Il lavoro viene svolto tramite il mouse e l'unico momento in cui si ricorre alla tastiera è quando occorre specificare il nome del file nel quale conservare il lavoro fatto. Disegnare icone, anche molto elaborate, è estremamente facile. Comunque, la dimostrazione migliore delle reali capacità del mouse è data dal secondo programma fornito a corredo: l'*AMX Art program*. Le funzioni di questo programma, un vero e proprio strumento software per la creazione di figure e immagini di qualsiasi genere, sono paragonabili a quelle del *MacPaint*, disponibile sul Macintosh. Anche l'impostazione generale richiama il programma della Apple.

Un programma artistico

Le immagini vengono create nella zona centrale dello schermo (il programma seleziona il MODE 4, con una risoluzione di 320 x 256 pixel in due colori). Lungo il margine destro dello schermo c'è il menu delle icone, che corrispondono alle varie azioni consentite (riempimento, cancellazione e via dicendo): il significato delle icone si spiega da solo. Una particolare azione viene selezionata portandovi sopra il cursore e premendo uno dei pulsanti di controllo del mouse: l'avvenuta selezione è confermata dal fatto che il cursore assume l'immagine dell'icona scelta. Il disegno si realizza muovendo il cursore (sempre tramite mouse, ovviamente) nella zona centrale dello schermo, dopo aver selezionato l'azione appropriata. Il funzionamento del sistema è di apprendimento talmente immediato, che raramente occorre consultare il manuale, redatto egregiamente.

Nella parte inferiore dello schermo compaiono altri due menu: uno contiene le indicazioni relative allo spessore o al tipo di linee da tracciare, mentre l'altro seleziona tutta una serie di "ombreggiature". Poiché la limitata memoria a disposizione consente l'uso di due soli colori, le ombreggiature sono l'unico sistema per dare profondità e forma ai disegni.

Tutta una serie di ulteriori funzioni (quali la LOAD e la SAVE delle immagini, l'accesso a comandi del sistema operativo e l'avvio del programma di copia su stampante dell'immagine creata, ad esempio) è contenuta in un gruppo di menu "avvolgibili", il cui nome compare in alto sullo schermo. Posizionando il cursore su uno di essi e premendo un pulsante di selezione, il menu si "srotola", coprendo momentaneamente parte del disegno realizzato, che ricompare quando il menu non serve più.

Dopo un po' di pratica, con questo programma si riesce a creare immagini di grande effetto che possono essere conservate e riprodotte in ogni momento.

Il programma è talmente valido che, da solo, giustifica pienamente l'acquisto dell'intero package. Tuttavia la AMX intende realizzare anche un package a basso costo, basato su quello qui esaminato, che assuma il totale controllo del Micro BBC, a guisa dei sistemi adottati sul Lisa e sul Macintosh (gestione generale, database ecc.). Per i possessori di BBC questa è un'opportunità da non perdere.

MOUSE AMX

DIMENSIONI

mm. 85 x 65 x 35

CPU

Un cavo consente il collegamento alla porta utente del Micro BBC

DOCUMENTAZIONE

Il package è corredato da due manuali, redatti con molta cura, che forniscono chiare informazioni sull'uso del sistema. Tuttavia, specialmente quello relativo all'AMX Art Program, vengono raramente consultati essendo il package facilissimo da usare

PRO

Il funzionamento è ottimo, i manuali anche: è un package che i possessori di BBC non dovrebbero lasciarsi sfuggire

CONTRO

La sfera metallica del mouse tende a slittare su superfici troppo lisce. È un peccato che esista la sola versione per Micro BBC

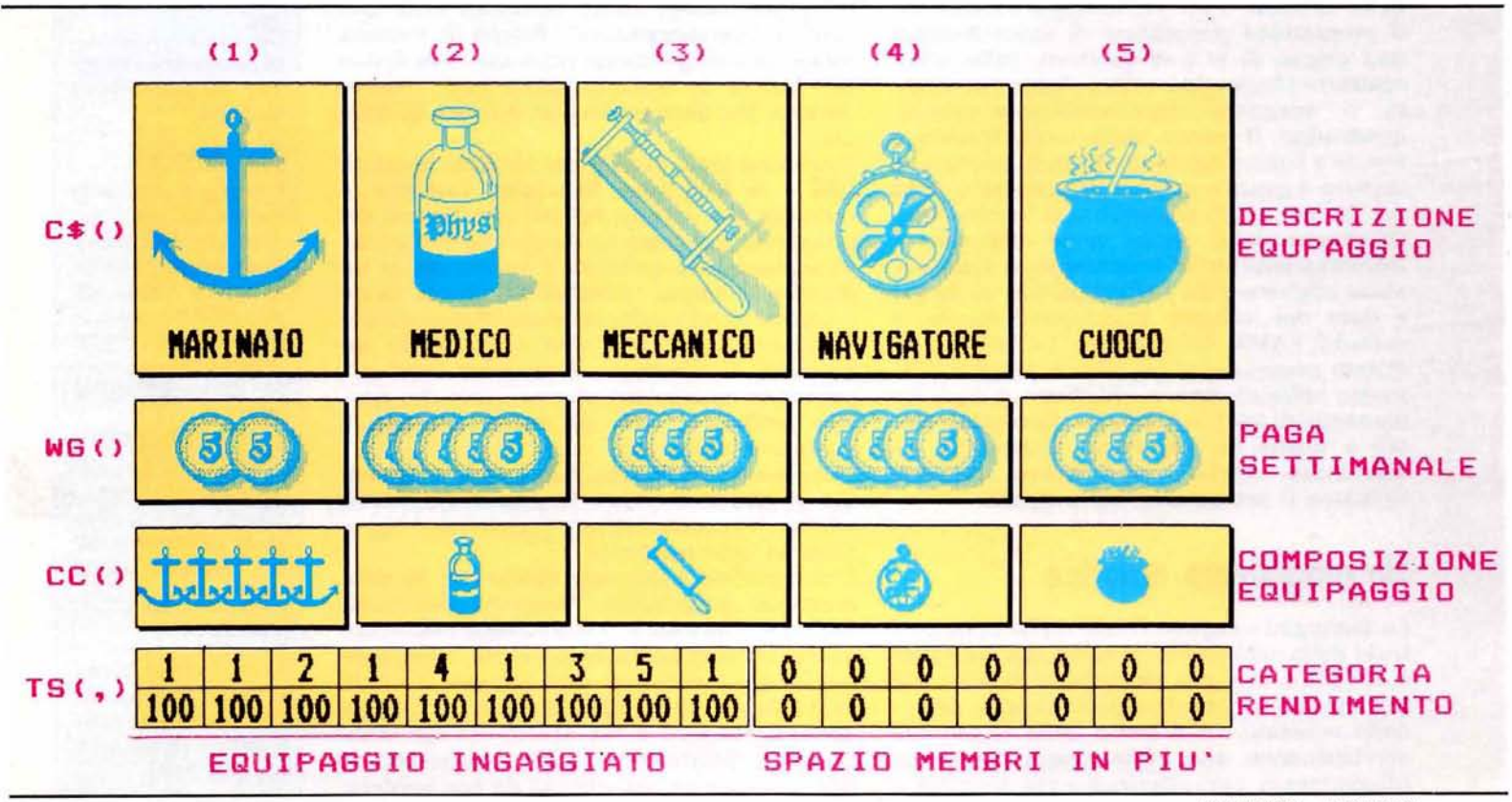
Tutti a bordo!

Esaminati i principi della simulazione, ecco un nuovo progetto di gioco d'avventura

In questo gioco di simulazione il giocatore è idealmente trasportato indietro nel tempo e assume il ruolo di un commerciante del XVI secolo, che intende organizzare una spedizione nel Nuovo Mondo. Occorre pianificare il viaggio, affrontarlo e, infine, concludere affari (magari con buon profitto!) con gli indigeni.
Il protagonista inizia il gioco come proprietario di un vascello e di 2000 monete d'oro da investire oculatamente. Il primo compito è quello di ingaggiare la ciurma, offrendo una paga settimanale per tutta la durata della spedizione: la paga dev'essere adeguata alle capacità di ciascun membro dell'equipaggio, che dev'essere composto da esperti di tutti i tipi (ma attenzione a non spendere più di quanto si possiede!). Il vascello prevede un equipaggio massimo di 16 uomini: ovviamente più sono i membri e maggiore è la somma da corrispondere, oltre al consumo di acqua e viveri.
Tuttavia, se l'equipaggio è ridotto al minimo, può non essere in grado di affrontare i numerosi pericoli di una lunga traversata (burrasche, attacchi di pirati, ecc.) allungando così il tempo necessario al viaggio e, di conseguenza, il costo dell'impresa.
Il gioco è redatto in BASIC, nella versione Microsoft, e progettato in modo che ogni nuovo modulo via via aggiunto possa funzionare indipendentemente. Il primo modulo provvede alla inizializzazione delle variabili e delle matrici necessarie al gioco e concerne anche l'ingaggio della ciurma. La prima sezione dimensiona alcune importanti matrici: C$(), ad esempio, contiene le informazioni relative alla ciurma. Sono previste cinque categorie di marinai, di conseguenza la matrice è dimensionata a 5 nella linea 16. Le singole descrizioni vengono immesse in ciascun elemento della matrice, cosicché basta usarne l'indice per risalire ad esse.
Le condizioni generali (ossia il rendimento) della ciurma variano durante il gioco, a seconda delle circostanze: per mantenerne la situazione aggiornata viene impiegata una matrice bidimensionale, TS(,), dimensionata in modo da avere 16 colonne e due righe. Ogni colonna rappresenta un membro dell'equipaggio, una riga si riferisce al tipo di mari-

Prospetto delle matrici

Per conservare le informazioni relative all'equipaggio ingaggiato per il viaggio vengono adoperate quattro matrici. C$( ), WG( ) e CC( ) contengono le caratteristiche associate a ciascuna categoria dell'equipaggio, rispettivamente: la descrizione, la paga settimanale e il numero di membri per categoria. TS(,) contiene invece la composizione dell'equipaggio che deve affrontare il viaggio, fino ad un massimo di 16 elementi. Ogni categoria dell'equipaggio viene contraddistinta da un numero progressivo (l'indice delle matrici), anziché usando il nome della categoria per esteso



IAN McKINNELL sul MACINTOSH

naio e l'altra al livello di rendimento, che inizialmente viene posto a 100.
L'importo delle paghe settimanali, suddiviso in cinque categorie, è conservato nella matrice a cinque elementi WG( ). Volendo esaminare la paga del medico di bordo, categoria 2, basta ispezionare WG(2). Le paghe settimanali sono: per i marinai semplici 10 monete d'oro, per i dottori 25, per i navigatori esperti 20, per i cuochi e per i meccanici 15. Per tener conto di quanto oro rimanga via via al giocatore viene usata la variabile MO. Questa viene inizializzata a 2000 nella linea 12 e in seguito decrementata, ad ogni settimana di viaggio, in base all'importo per i salari e alle spese sostenute. Via via che il gioco va avanti occorre sapere quanti membri dell'equipaggio appartengono a ciascuna delle cinque categorie (per calcolare la loro efficienza in particolari momenti critici). Queste informazioni potrebbero essere di volta in volta desunte dalla riga superiore della matrice TS, ma è più comodo riunirle in un'apposita matrice CC( ), che viene dimensionata a 5 nella linea 18.

Informazioni preliminari

Avendo inizializzato le variabili e le matrici, il modulo di programma passa a visualizzare la durata approssimativa del viaggio e l'ammontare di denaro che conviene accantonare per ogni eventualità. La visualizzazione non avviene nel modo consueto: una speciale subroutine, alla quale viene passato ciascun messaggio da far comparire, visualizza ogni carattere della scritta simulando il comportamento di una telescrivente. La subroutine che si occupa di ciò, alla linea 9100, spezzetta la frase passata nella variabile TS$, visualizzandone un carattere alla volta. Una seconda subroutine, alla linea 9200, inserisce pause di maggior durata (quando necessario) tra una riga di scritto e quella successiva.
La subroutine alla linea 1000 consente di selezionare l'equipaggio, digitando un numero da 1 a 5 (il tasto F per Finire). Le immisioni valide vengono conservate in TS(,) dalla linea 1156, mentre CN tiene conto del numero di membri via via ingaggiati. Ad ogni nuova assunzione, viene visualizzato il totale dei salari settimanali, WT, calcolato sommando l'opportuno elemento di WG( ) al precedente valore di WT (vedere la linea 1158). Durante ogni fase, la sezione di programma tra le linee 1160-1190 visualizza la composizione dell'equipaggio. La linea 1185 controlla se il numero di membri, per ogni categoria, è maggiore di zero o di uno, accodando la lettera terminale adatta alla relativa descrizione.
In varie occasioni, durante il gioco, viene chiesto premere un tasto qualsiasi per continuare. Un messaggio in tal senso, contenuto in K$, viene trasferito per la visualizzazione in S$, quando occorre. Infine, terminate le operazioni di ingaggio, il modulo termina con un RETURN. Il prossimo modulo esaminato è quello relativo all'acquisto di derrate e materiale vario.

MODULO UNO

Il primo modulo del gioco d'avventura inizializza le variabili e consente l'ingaggio della ciurma, fino a un massimo di 16 uomini, da scegliere tra le cinque categorie disponibili: marinai, medici, meccanici, navigatori e cuochi. Le informazioni vengono conservate in matrici di adeguate dimensioni. L'intero programma è scritto nella versione BASIC Microsoft, quindi si adatta alla maggior parte dei microcomputer (le modifiche necessarie per alcuni apparecchi sono riportate in "Basic & Basic")

```
9 K$ = "   PREMI QUALSIASI TASTO..."
10 DIM TS(16,2):REM CATEGORIA/RENDIMENTO DELLA CIURMA
11 CN=0:     REM NUMERO COMPONENTI DELLA CIURMA
12 MO=2000: REM MONETE D'ORO INIZIALI
13 DIM WG(5):WG(1)=10:WG(2)=25:WG(3)=15:WG(4)=20:WG(5)=15:REM PAGHE
14 WT=0:     REM TOTALE PAGHE SETTIMANALI
15 CM=16:    REM NUMERO MASSIMO MEMBRI
16 DIM C$(5):C$(1)="MARINAI":C$(2)="MEDICI":C$(3)="MECCANICI"
17 C$(4)="NAVIGATORI":C$(5)="CUOCHI"
18 DIM CC(5):REM CONTO PER CIASCUNA CATEGORIA
80 PRINT CHR$(147):S$="  COMMERCIO NEL NUOVO MONDO#":GOSUB 9100:PRINT
81 GOSUB 9200
82 S$="SEI IL CAPITANO DI UN VASCELLO.#":GOSUB 9100:PRINT
83 S$="INTENDI NAVIGARE VERSO IL NUOVO#":GOSUB 9100:PRINT
84 S$="MONDO. DI SOLITO IL VIAGGIO DURA#":GOSUB 9100:PRINT
85 S$="8 SETTIMANE, MA PUO' RICHIEDERE#":GOSUB 9100:PRINT
86 S$="PIU' TEMPO. DEVI INGAGGIARE LA#":GOSUB 9100:PRINT
87 S$="CIURMA, PAGARLA E ACQUISTARE#":GOSUB 9100:PRINT
88 S$="MATERIALE VARIO PER IL COMMERCIO.#":GOSUB 9100:PRINT
89 S$="POSSIEDI 2000 MONETE D'ORO.#":GOSUB 9100:PRINT:GOSUB 9200
90 S$="        BUONA FORTUNA!!!#":GOSUB 9100:PRINT:GOSUB 9200:PRINT
91 S$=K$:GOSUB 9100
94 GET P$:IF P$="" THEN 94
95 GOSUB 9200
500 GOSUB 1000
999 END
1000 PRINT CHR$(147):PRINT "FASE 1 - INGAGGIO CIURMA"
1010 PRINT "-------------------------"
1012 PRINT
1015 GOSUB 9200
1020 PRINT:PRINT"CATEGORIE DI EQUIPAGGIO:"
1025 GOSUB 9200
1030 PRINT
1040 PRINT"TIPO DESCRIZIONE      PAGA SETTIMANALE"
1050 PRINT"---- -----------      ----------------"
1060 PRINT"  1  MARINAI           10 MONETE D'ORO"
1070 PRINT"  2  MEDICI            25 MONETE D'ORO"
1080 PRINT"  3  MECCANICI         15 MONETE D'ORO"
1090 PRINT"  4  NAVIGATORI        20 MONETE D'ORO"
1100 PRINT"  5  CUOCHI            15 MONETE D'ORO"
1105 GOSUB 9200
1110 PRINT:PRINT:PRINT
1120 S$="QUALE TIPO SCEGLI (1-5)#":GOSUB 9100
1122 S$="(F = FINE INGAGGIO)#":GOSUB 9100:PRINT:INPUT P$
1125 CT=VAL(P$)
1128 IF LEFT$(P$,1)="F" THEN PRINT:PRINT"FINE INGAGGIO.":GOSUB 9200:GOTO 1310
1130 IF CT>0 AND CT<6 THEN 1150
1139 PRINT:PRINT
1140 PRINT P$;:S$="  NON E' UN TIPO PREVISTO!#":GOSUB 9100
1142 GOSUB 9200
1145 S$="RIPETI IL COMANDO...#"
1146 GOSUB 9100
1147 GOTO 1300
1150 PRINT:PRINT
1155 CN=CN+1:REM NUMERO DI COMPONENTI LA CIURMA
1156 TS(CN,1)=CT: REM CATEGORIA SCELTA
1157 TS(CN,2)=100:REM RENDIMENTO INIZIALE
1158 WT=WT+WG(CT):REM TOTALE PAGHE
1159 CC(CT)=CC(CT)+1:REM CONTO CATEG. CIURMA
1160 S$="CIURMA ATTUALE:"
1170 FOR T=1 TO 5
1180 PRINT S$;CC(T);" ";C$(T);
1189 S$="          "
1190 NEXT
1195 PRINT:PRINT"TOTALE PAGHE SETTIMANALI ";WT
1200 IF CN=CM-1 THEN PRINT:S$="ULTIMO MEMBRO#":GOSUB 9100:GOTO1295
1202 IF CN=CM THEN PRINT :S$="  EQUIPAGGIO COMPLETO!!#":GOSUB9100:GOTO1310
1300 GOTO 1015
1310 PRINT:S$=K$:GOSUB 9100:PRINT:GOSUB 9200
1320 GET P$:IF P$="" THEN 1320
1990 RETURN
9100 REM VISUALIZZAZIONE RALLENTATA
9110 FOR S3 = 1 TO 32
9115 IF MID$(S$,S3,1)="#" THEN S3=32:GOTO 9140
9120 PRINT MID$(S$,S3,1);
9130 FOR S4 = 1 TO 25:NEXT
9140 NEXT:PRINT
9199 RETURN
9200 REM CICLO DI RITARDO LUNGO
9210 FOR S5 =1 TO 1000:NEXT
9299 RETURN
```

Basic & Basic

Spectrum:

Apportare le seguenti modifiche:

```
16 DIM C$(5,9):C(1)$ = "MARINAIO":C$(2) = "MEDICO":C$(3) = "MECCANICO"
80 CLS:LET S$ = "COMMERCIO NEL NUOVO MONDO":GOSUB 9100:PRINT
94 LET P$ = INKEY$:IF P$ = "" THEN GOTO 94
1005 CLS:PRINT"FASE 1 - INGAGGIO CIURMA"
1128 IF P$(1 TO 1) = "F" THEN PRINT "FINE INGAGGIO.":GOSUB 9200:GOTO 1310
9115 IF S$(S3 TO S3) = "*" THEN S3 = 32:GOTO 9140
9120 PRINT S$(S3 TO S3);
```

Micro BBC:

Apportare le seguenti modifiche:

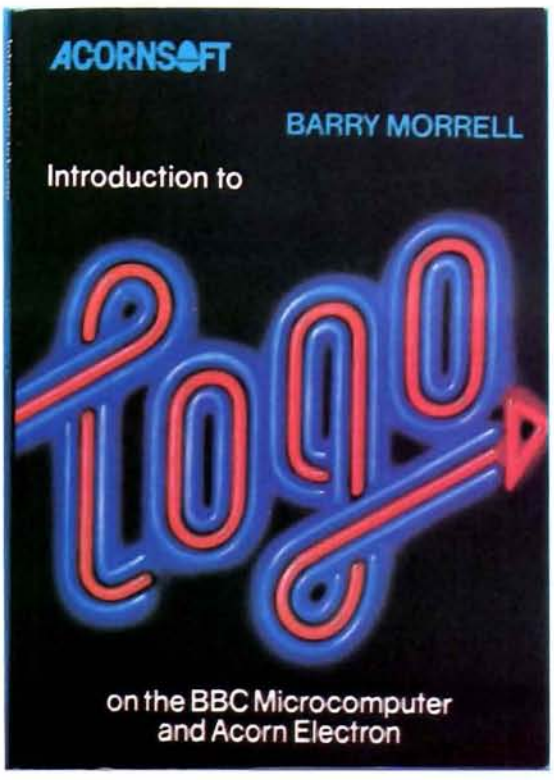
```
80 CLS:S$ = "COMMERCIO NEL NUOVO MONDO":GOSUB 9100:PRINT 94 P$ = GET$
1005 CLS:PRINT"FASE 1 - INGAGGIO CIURMA"
```

Tartarughe... scolastiche

Un esame di due ottimi package di LOGO per il BBC: le versioni Acornsoft e Logotron

Il LOGO della Acornsoft

Il linguaggio LOGO ha incontrato il favore di molti insegnanti nelle scuole, che lo ritengono un ottimo strumento didattico. In Gran Bretagna, dove il Micro BBC è diffusissimo nelle scuole, sono stati realizzati diversi package di LOGO.
Le due versioni che sembrano dominare il mercato, comunque, sono quella prodotta dalla Acornsoft e quella della Logotron, entrambe fornite su chip ROM.
Lo schermo grafico misura circa 1200 x 800 passi di tartaruga, offrendo una risoluzione migliore di molti altri pacchetti di LOGO. Tutti gli otto MODE del BBC sono disponibili, così come una vasta gamma di tracciamenti grafici (linee tratteggiate, riempimenti triangolari, ecc.). Se una particolare applicazione non richiede grandi doti grafiche conviene selezionare il MODE 7, che lascia un'ampia fetta di memoria disponibile per il programma. Per quanto concerne l'audio, sono previsti due comandi: SOUND e ENVELOPE, che funzionano in modo identico ai loro equivalenti in BASIC.



Per editare le linee di programma c'è il consueto comando COPY, oltre alla possibilità di usare i codici di controllo del BBC, i comandi VDU, *FX e quelli *. Entrambe le versioni, a detta delle case produttrici, sono capaci di lavorare anche con il secondo processore (6502) e con la rete Econet.
La versione Acornsoft, scritta e compilata in BCPL, fornita su due chip ROM da 16 Kbyte, è accompagnata da tre libri, un nastro a cassetta e un dischetto (che contengono programmi dimostrativi e di utilità) e possiede, oltre a varie funzioni extra, ben 200 "primitive", un numero decisamente alto rispetto alla concorrenza.
Vediamo quali sono le più importanti funzioni extra. Un gruppo di primitive consente di modificare l'ambiente operativo del LOGO, per adattarlo alle esigenze scolastiche. Due caratteristiche sono particolarmente utili: la prima permette di ridefinire le primitive e di "seppellire" le procedure in modo che appariano come vere primitive; la seconda permette di preparare alcuni file, da poter avviare automaticamente all'accensione del computer.
Per facilitare l'individuazione e la correzione degli errori esistono varie opzioni trace: su ogni linea e su ogni chiamata a una procedura oppure a una primitiva, con la possibilità di fermarsi ad ogni passo.
Ciò rappresenta una notevole innovazione, molto utile anche per le spiegazioni sulla logica di programmazione. I comandi CATCH e THROW consentono di saltare da una procedura all'altra.
Queste primitive servono soprattutto per la cattura di errori nei parametri in ingresso, assieme a molti altri strumenti offerti dal package.
Tra le estensioni del linguaggio, conservate su dischetto, c'è un gruppo di primitive, di diretta discendenza dal LISP, per la gestione delle "proprietà" (presenti anche nel LOGO LCSI per l'Apple). Un altro gruppo di primitive offre la possibilità di creare fino a 32 tartarughe simultaneamente, ognuna con una propria forma e indirizzabile singolarmente. Esistono poi estensioni per consentire l'uso di specifiche tartarughe da pavimento (la Buggy, la Jessop e la Valiant).

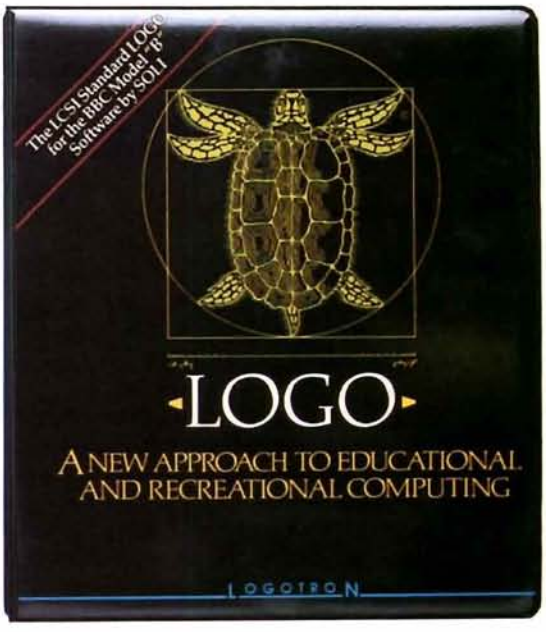
Il LOGO della Logotron

Questa versione, che aderisce allo standard

LCSI, è stata scritta in codice macchina e viene fornita su di un unico chip ROM da 16 Kbyte, accompagnato da un manuale descrittivo e di consultazione. Questo è stato sviluppato dalla SOLI (Systémes d'Ordinateur Logo International), un organismo che ha collaborato in passato con la LCSI e ha prodotto la versione LOGO per lo Spectrum.
La SOLI ha dotato questo LOGO di circa 120 primitive, nell'evidente tentativo di racchiudere il tutto in un unico chip di ROM, in modo da velocizzare il funzionamento, senza occupare più zoccoli di ROM. La versione Logotron ha un gruppo di caratteristiche degne di rilievo, una delle quali permette di interagire con programmi scritti su file (come nel LOGO Atari).
Molta cura è stata prestata nella progettazione dello screen editor, che è controllato tramite i tasti funzione e le opzioni FIND (cerca) e REPLACE (sostituisci).
Questa versione contiene anche due nuove primitive: OPPS e OPNS, che producono un elenco dei nomi, rispettivamente, delle procedure e delle variabili. Queste primitive rappresentano una valida alternativa alle consuete POTS e PONS.
Una caratteristica molto interessante è la possibilità di passare variabili ai comandi del sistema operativo del BBC. Ciò consente di redigere procedure dal nome appropriato e comprensibile anziché dai nomi oscuri richiesti dal SO del BBC.
Brillano, per la loro assenza, le primitive TEXT e DEFINE. Nel manuale è spiegato come ottenerle tramite definizioni, ma dato che operano su testi conservati su file, il loro funzionamento è necessariamente lento. I più ligi alla programmazione strutturata saranno lieti di sapere che questa versione non prevede l'istruzione GO. Il comando TRACE visualizza il nome di ciascuna procedura eseguita, assieme ai parametri in ingresso, ma nessun messaggio viene prodotto alla sua conclusione: un comando TRACE più completo sarebbe certamente più utile.

Quale tra i due?

Sui due prodotti sono stati eseguiti alcuni test, per esaminare le rispettive caratteristiche in termini di velocità d'esecuzione e occupazione di memoria. La versione LCSI ha un poi più spazio per la definizione di variabili e procedure e un'area più ampia riservata allo stack (quasi il doppio), consentendo così un maggior numero di livelli di ricursione, prima di esaurire la memoria a disposizione.
Dopo una serie di test sulla velocità di esecuzione, si è rilevato che la versione Acornsoft impiega dal 50% al 1000% in più rispetto alla versione Logotron: i massimi si toccano nell'elaborazione di liste e nelle assegnazioni. Ciò è probabilmente dovuto alla lentezza del "garbage collector" (la routine di sistema per la periodica riorganizzazione della memoria) della versione Acornsoft.
La lentezza di questa versione, nell'elaborazione di liste, è a dir poco esasperante. Tuttavia sul dischetto, assieme al LOGO, viene fornito un linguaggio simil-PROLOG, scritto in LOGO, che funziona in modo abbastanza rapido.
Entrambe le versioni sono corredate da manuali ricchi di dettagli, che contengono anche una sezione didattica sul linguaggio dedicata ai principianti e una sezione di consultazione. Viene ampiamente spiegata e illustrata la "geometria della tartaruga", benché nella versione Acornsoft talvolta non si riesca a comprendere bene la distinzione tra comando e operazione. Entrambi i manuali dedicano appena 8 pagine alla descrizione dell'elaborazione di liste, ma nella versione Logotron si cerca in parte di colmare questa lacuna fornendo diversi listati di procedure che aiutano a comprendere e a semplificare tale argomento. Queste procedure sono il frutto del lavoro di Mike Sharples presso il Dipartimento di Intelligenza Artificiale dell'Università di Edimburgo.
Anche la Acornsoft fornisce, su disco o su cassetta, dei programmi dimostrativi (alcuni dei quali scritti in simil-PROLOG), oltre a qualche esempio di analisi sul linguaggio naturale.
Per la versione Acornsoft sono state annunciate future espansioni, di cui non vengono però forniti dettagli. La Logotron sta per mettere in commercio una versione "espansa" del LOGO, su disco, che dovrebbe aggiungere numerose altre primitive al linguaggio; ancor più interessante è l'annuciato lancio, sempre da parte della Logotron, di una scheda hardware che consentirà di adoperare ben 30 sprite. Detto per inciso, la Logotron afferma che il proprio software è "a prova di pirati", ossia non è copiabile. Eventuali ROM danneggiate vengono rimpiazzate con modica spesa.



Interazioni del BASIC

Quali rapporti esistono tra il BASIC e il sistema operativo del BBC?

In una parte precedente (vedere a pag.1338) si è detto come il sistema operativo possa fornire una protezione nei confronti delle modifiche alla configurazione hardware di un computer. Non deve sorprendere, quindi, se il BASIC del BBC fa largo impiego delle routine del SO. Il comando SOUND del BASIC, ad esempio, genera una chiamata OSWORD con A=7 (vedere a pag. 1458) per produrre suoni. Similmente, ENVELOPE genera una chiamata OSWORD con A=8.
È interessante notare che, modificando alcune routine del SO, è possibile alterare il comportamento del BASIC. Basta intervenire sugli indirizzi di alcuni vettori del sistema, per dirigerli verso routine in codice macchina realizzate dall'utente.
In modo molto simile, anche la chiamate OSWRCH, OSACII e OSNEWL vengono adoperate dalle routine del BASIC che gestiscono lo schermo.
Va detto, comunque, che talvolta l'uso di tali chiamate in programmi per la grafica, scritti in codice macchina, non produce una più rapida esecuzione rispetto agli equivalenti comandi in BASIC. Ciò dà una misura della particolare efficienza del BASIC nel manipolare le routine del sistema operativo in comandi quali MOVE, DRAW e PLOT.
Vediamo adesso i due comandi BASIC che consentono di richiamare routine scritte in codice macchina da un programma BASIC. Questi comandi sono: USR e CALL.

La chiamata USR

La funzione dell'istruzione USR è quella di consentire l'esecuzione di una routine in codice macchina al termine della quale il controllo ritorna al BASIC con precise indicazioni sul contenuto dei registri A, X e Y del microprocessore, oltre al registro di stato (PSR) e del flag di carry. Il formato della chiamata USR è il seguente:

```
result% = USR (indirizzo%)
```

dove indirizzo% è l'indirizzo della routine da eseguire. Prima della chiamata è consentito depositare valori nelle variabili A%, X% e Y% (corrispondenti rispettivamente ai registri A, X e Y). Il bit meno significativo di C% può essere usato per alterare il flag di carry (un qualsiasi numero pari ad esso assegnato azzera il flag). Il seguente programma fornisce un semplice esempio sull'uso di USR, in cui viene chiamata la routine OSBYTE che ritorna con le coordinate x e y del cursore testuale rispettivamente nei registri X e Y. Il valore di risultato% rappresenta il valore di tali registri, ma in una forma codificata che viene esaminata più avanti.

```
10 A% = 134
20 X% = 0
30 Y% = 0
40 LET risultato% = USR (&FFF4)
50 PRINT risultato%:REM in forma esadecimale
```

LA FUNZIONE USR IN AZIONE

La funzione USR permette di eseguire, in BASIC, una routine in codice macchina progettata in modo da fornire valori di ritorno. Poiché la USR è una funzione (la CALL è invece un comando) che fornisce un valore, questo risultato va assegnato internamente al programma BASIC, o sotto forma di variabile:

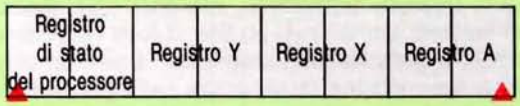
risultato% = USR (&FFF4)

o come parte di un'istruzione PRINT:

PRINT USR (&FFF4)

Il risultato riportato in BASIC dalla funzione USR è contenuto in una variabile intera composta da quattro byte, che riflette il contenuto dei quattro registri P, Y, X e A secondo questo formato:

risultato% =



Byte più significativo — Byte meno significativo

Il valore desiderato può essere facilmente estratto ricorrendo a una AND per mascherare i bit che non interessano. Ad esempio:

result% AND &000000FF

elimina, filtrandoli, tutti i valori eccetto quello del registro A.

Visualizzare il risultato in forma esadecimale facilita l'interpretazione dei valori forniti dalla chiamata OSBYTE.

Il comando CALL

Il metodo alternativo, per eseguire routine in codice macchina, è dato dal comando CALL, usato più volte in precedenza (in particolare per chiamare OSWORD e altre routine del SO). In tali occasioni la chiamata conteneva semplicemente un riferimento all'indirizzo della routine da eseguire, secondo il formato:

```
CALL &FFF1
```

Il BASIC del BBC, tuttavia, prevede una versione estesa di questo comando, che consente di passare valori contenuti in variabili del programma BASIC alla routine in codice

macchina. Il problema maggiore, in questo caso, consiste nel far sapere al programma in codice macchina dove, in memoria, trovare i valori trasmessi. Fortunatamente, di ciò si occupa lo stesso BASIC. In una chiamata CALL accompagnata da parametri, quale:

```
CALL indirizzo%,A%,C%
```

tutti i particolari relativi ai parametri (A% e C% in questo caso) vengono depositati nella pagina 6 della memoria, conosciuta appunto come "blocco dei parametri". Ecco la struttura di tale blocco:

Locazione	Descrizione del contenuto
&0600	Numero di parametri
&0601	Indirizzo del primo parametro
&0602	
&0603	Tipo del primo parametro
&0604	Indirizzo del secondo parametro
&0605	
&0606	Tipo del secondo parametro

Ovviamente, le informazioni circa ciascun elemento vengono ripetute tante volte quanti sono i parametri passati. L'indirizzo del parametro è semplicemente quello in cui si trova, in memoria, il primo byte del parametro. Il valore del "tipo" si riferisce al genere di variabile impiegata e può essere dedotto dalla seguente tabella:

Tipo	Descrizione	Esempio
0	Byte (8 bit)	CALL indirizzo, ?&70
4	Variabile intera	CALL indirizzo, F%
8	Variabile reale	CALL indirizzo, G
128	Stringa (con indirizzo)	CALL indirizzo, $(&A00)
129	Variabile stringa	CALL indirizzo, F$

L'ultimo elemento, parametro di tipo 129, è leggermente diverso dagli altri, in quanto il suo indirizzo nel blocco dei parametri non è l'effettivo indirizzo della stringa, bensì quello della zona di memoria che contiene ulteriori informazioni sulla stringa (chiamata, appunto, "blocco di informazioni sulle stringhe"), così strutturata:

Locazione	Descrizione del contenuto
Indirizzo nel blocco dei parametri	Indirizzo della stringa
	Numero di byte assegnati
	Lunghezza della stringa

Pertanto, i primi due byte del blocco di informazioni contengono l'indirizzo del primo carattere della stringa. Il numero di byte assegnati corrisponde alla massima lunghezza fissata per la stringa, la cui lunghezza effettiva si trova nel quarto byte.

Passaggio di parametri

Il comando CALL consente di eseguire una sezione di codice macchina dall'interno di un programma BASIC ma, a differenza di USR, non ritorna con dei risultati. Tuttavia, l'utente può far sì che la stessa routine in codice macchina depositi alcuni valori in varibili del programma BASIC, in seguito accessibili a quest'ultimo.
Il seguente programma non solo fornisce un esempio di chiamata CALL con passaggio di parametri, ma deposita anche il risultato ottenuto nella variabile A%. Per di più, esso dimostra come si possono generare messaggi d'errore all'interno di una routine in codice macchina, ricorrendo alla routine di sistema per il trattamento degli errori (vedere a pag. 1518). È scritto in BASIC II:

```
 10 DIM C 400
 20 zero = &70
 30 FOR I% = 0 TO 2 STEP 2
 40 P% = C
 50 [OPT I%
 60 .codice LDA &600
 62 CMP #1
 64 BNE errore
 70 LDA &603
 80 CMP #4
 90 BNE errore2
100 LDA &601:STA &70
110 LDA &602:STA &71
120 LDY #0
130 LDA &02:STA (zero),Y:INY
140 LDA &03:STA (zero),Y
160 RTS
170 .error BRK
180 EQUB 254
190 EQUS "Numero di parametri errato!"
200 EQUB 00
210 .err2 BRK
220 EQUB 253
230 EQUS "Tipo non corrisponde!"
240 EQUB 00
250 ]:NEXT
260 A% = 0:CALL codice,A%
270 PRINT "Indirizzo finale variabili BASIC";A%
```

Questo programma passa nella variabile intera A% l'indirizzo terminale dell'area delle variabili. Ciò dimostra che il passaggio di parametri è possibile anche con la chiamata CALL. Le linee 60-64 verificano che sia presente un solo parametro, altrimenti viene visualizzato l'apposito messaggio d'errore. Le linee 70-90 verificano quindi che il parametro passato sia di tipo intero, altrimenti viene emesso il secondo messaggio d'errore. Le linee 100-160 trasferiscono il contenuto dell'indirizzo 2 e 3 nei byte meno significativi della variabile intera usata come parametro. Le linee 170-240 si occupano di visualizzare gli eventuali messaggi d'errore.
La CALL si trova nella linea 260. La variabile A% è azzerata prima della chiamata, per esser certi che non contenga valori indesiderati nei byte più significativi. Dopo l'esecuzione, digitando HIMEM-A% si ottiene il numero di byte rimasti liberi, tenendo conto dell'occupazione dovuta al programma BASIC e alle variabili usate.

A me gli occhi!

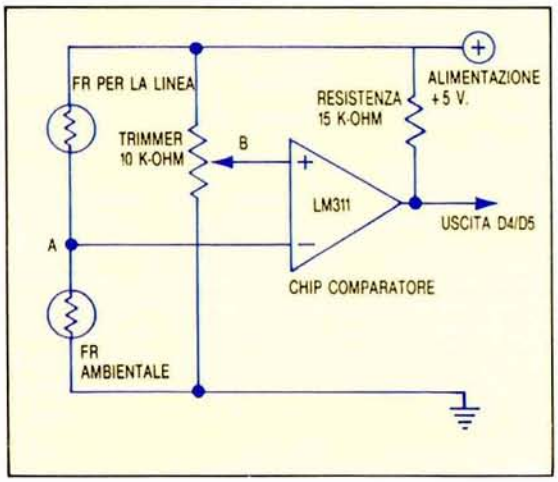
Dotiamo il nostro veicolo robot di rudimentali organi di vista

L'unità qui descritta, che consente al robot di seguire una linea tracciata sul pavimento, consiste in due circuiti identici, ciascuno dei quali invia un proprio segnale alla porta utente. Ciascun circuito è realizzato attorno ad un circuito integrato LM311, che confronta le tensioni presenti su due suoi piedini. Se la tensione al piedino "positivo" è maggiore di quella applicata al piedino "negativo", l'uscita passa immediatamente da zero alla tensione di alimentazione. In termini logici, l'uscita di questo chip è 0 se le due tensioni in ingresso sono identiche, altrimenti, se il voltaggio al piedino positivo supera quello del piedino negativo, è 1.

Due fotoresistenze (FR) in ciascun circuito formano un partitore di tensione tra voltaggio di alimentazione e massa. Se la quantità di luce che colpisce ciascuna fotoresistenza è uguale, nel punto A del circuito troviamo una tensione di circa 2,5 volt, pari a circa la metà di quella di alimentazione. Il trimmer (o potenziometro semifisso) va regolato in modo da ottenere nel punto B una tensione identica a quella del punto A. In parità di luce, quindi, il voltaggio presente ai due ingressi è identico e l'uscita dal chip vale zero. Se però la luce che cade sulla FR della linea diminuisce (passando, ad esempio, su di una striscia scura), la sua resistenza aumenta e il voltaggio nel punto A supera i 2,5 volt. Ciò provoca il passaggio dell'uscita da zero al voltaggio di alimentazione (livello logico 1), indicando che la FR si trova sopra una zona scura.

La presenza di due circuiti è necessaria per individuare da quale lato della linea si sta scostando il robot.

Delle due FR per ogni circuito, una misura l'intensità della luce ambientale per confrontarla con la lettura dell'altra che è orientata sulla linea da seguire.



Il circuito

Qui a lato è riportato lo schema elettrico del circuito che consente al robot di seguire una linea nera tracciata sul pavimento. Occorrono due circuiti identici a questo, uno per ogni lato della traccia pilota. La luce captata da ciascuna delle due fotoresistenze (FR) viene convertita in tensione e i due segnali risultanti vengono confrontati dal chip LM311. L'uscita vale zero solo se i due segnali risultano uguali

Taratura del progetto

Dopo aver costruito il dispositivo, occorre calibrarne il funzionamento. In condizioni di luce ambientale simili a quelle in cui si desidera far funzionare il robot, si lanci il seguente programma di taratura.

I PROGRAMMI DI CALIBRAZIONE DELLE FR

```
1000 REM **** VERIFICA FR - BBC ****
1010 DDR=&FE62:REGDAT=&FE60:?DDR=15:REM LINEE 0-3 IN USCITA
1020 REPEAT
1030 contenuto=?REGDAT
1040 IF(contenuto AND 16)=0 THEN PRINT TAB(5) "SINISTRA";
1050 IF(contenuto AND 32)=0 THEN PRINT TAB(15) "DESTRA";
1060 PRINT
1070 UNTIL FALSE

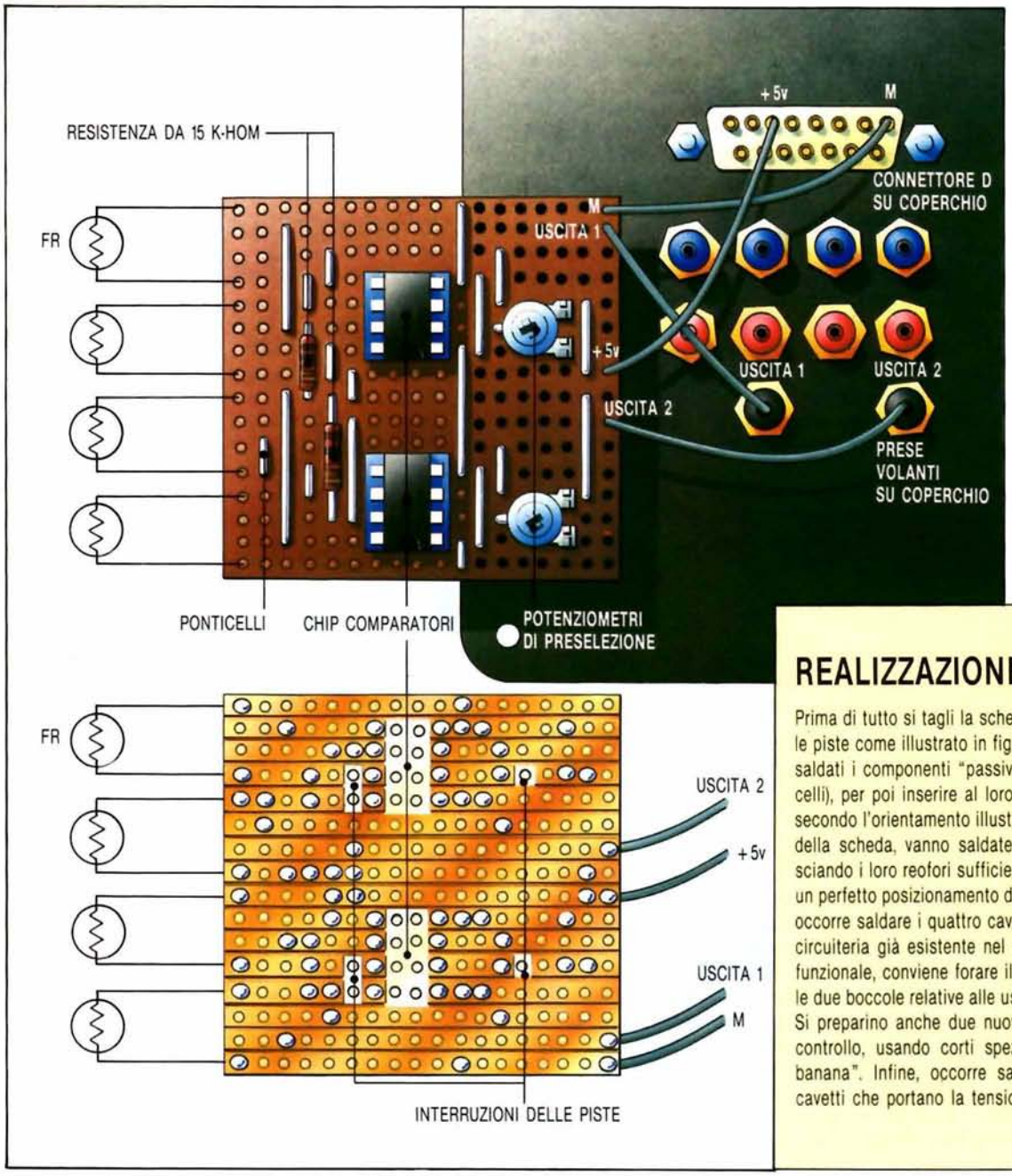
1000 REM **** VERIFICA FR - CBM ****
1010 DDR=56579:REGDAT=56577:POKE REGDAT,15
1020 CN=PEEK(REGDAT)
1030 IF(CN AND 16)=0 THEN PRINTTAB(5)"SINISTRA";
1040 IF(CN AND 32)=0 THEN PRINTTAB(15)"DESTRA";
1050 PRINT
1060 GOTO 1020

1000 REM **** VERIFICA FR - SPECTRUM ****
1010 CLEAR 32499:LET ST=32500: GO SUB 3000
1020 LET NM=16:GO SUB 2000:IF USR ST=0 THEN PRINTTAB 5;"SINISTRA";
1030 LET NM=32:GO SUB 2000:IF USR ST=0 THEN PRINTTAB 15;"DESTRA";
1040 PRINT
1050 GO TO 1020
1060 :
2000 REM **** ESEGUE AND ****
2010 POKE ST+1,IN 31
2020 POKE ST+3,NM:RETURN
2030 :
3000 REM **** CARICATORE IN CODICE MACCHINA ****
3010 FOR I=ST TO ST+8
3020 READ A:POKE I,A
3030 NEXT I
3040 DATA 62,0,14,0,161,6,0,79,201
3050 RETURN
```

Si prenda un foglio bianco e vi si tracci una riga nera spessa 2,5 centimetri. Adesso, a turno, si agisca sui due trimmer in modo che l'uscita di ciascun circuito sia zero, quando entrambe le FR centrali si trovano sulla riga

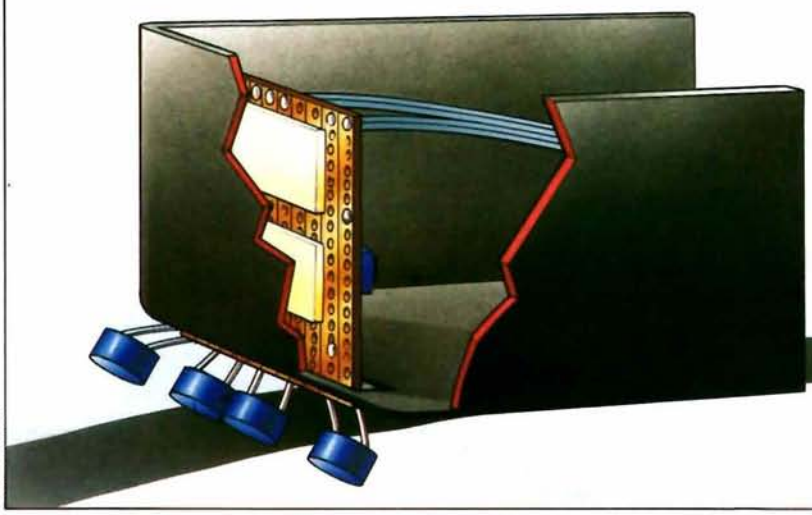
ELENCO COMPONENTI

Quantità	Descrizione
2	chip comparatori LM311
2	prese DIL a 8 piedini
2	prese da 2mm.
2	resistenze da 15 KOhm, 1/4 Watt
2	trimmer orizzontali da 10 KOhm
4	fotoresistenze ORP 12
1	scheda tipo veroboard 50 fori/24 piste
1	cuscinetti autoadesivi
1 m	cavo a nastro (4 vie)
1 m	stagno per saldature



REALIZZAZIONE PRATICA

Prima di tutto si tagli la scheda del circuito, interrompendo le piste come illustrato in figura. Per primi vanno montati e saldati i componenti "passivi" (resistenze, zoccoli e ponticelli), per poi inserire al loro posto i due circuiti integrati, secondo l'orientamento illustrato in figura. Infine, sul bordo della scheda, vanno saldate le quattro fotoresistenze, lasciando i loro reofori sufficientemente lunghi da consentire un perfetto posizionamento delle FR. Completata la scheda, occorre saldare i quattro cavetti per il collegamento con la circuiteria già esistente nel robot. Per rendere il tutto più funzionale, conviene forare il coperchio del robot e fissarvi le due boccole relative alle uscite dei due chip comparatori. Si preparino anche due nuovi ponticelli per il pannello di controllo, usando corti spezzoni di filo e due prese "a banana". Infine, occorre saldare al connettore D i due cavetti che portano la tensione di alimentazione



KEVIN JONES

GLI "OCCHI" DEL ROBOT

Il circuito che contiene gli "occhi" va montato nella scatola del robot come illustrato in figura. Le fotoresistenze devono sporgere adeguatamente ed essere orientate in modo corretto, facendo attenzione a non danneggiare i loro terminali. Le due fotoresistenze centrali vanno accostate il più possibile, in modo da ridurre al minimo lo spessore della banda nera che serve come traccia di guida

nera. Spostando leggermente il veicolo a destra o a sinistra, l'uscita deve immediatamente passare al livello 1. Se tutte le FR si trovano sulla zona bianca del foglio, sullo schermo deve comparire la scritta SINISTRA DESTRA. Adesso, muovendo lentamente il robot da sinistra a destra, fino a giungere sulla traccia nera, sullo schermo deve apparire la scritta SINISTRA, che scompare quando entrambe le FR centrali si trovano sulla banda nera. Proseguendo nel movimento, deve apparire la scritta DESTRA e, infine, la scritta SINISTRA DESTRA, quando tutte le FR si trovano nuovamente sulla zona bianca. Il progamma presume che la FR di sinistra sia collegata alla linea 4 e quella di destra alla linea 5.
L'illuminazione ambientale deve rimanere costante durante la taratura e durante l'uso del robot, per cui conviene orientare sulla zona interessata ai movimenti del robot una serie di lampade (meglio se di una certa potenza) onde poter ricreare a piaciemento condizioni costanti di illuminazione.

Il programma "inseguitore"

Il programma che si occupa di far seguire la traccia al robot adopera le routine di movimento già viste in precedenza. Si suppone che il robot parta perfettamente centrato sulla banda nera. All'interno di un ciclo iterativo il programma fa muovere il robot di un 1mm. e poi verifica se esso si è scostato dalla riga nera (evento segnalato quando i bit 4 o 5 producono un livello logico zero). Se, per esempio, è il bit 4 che passa a zero, il robot si è scostato verso sinistra, cosicché il programma rimedia eseguendo una piccola rotazione di 5° verso destra, prima di proseguire. L'effetto opposto si ha se è invece il bit 5 che passa a zero. Il programma termina quando entrambi i bit passano a zero, segnalando che la striscia nera è finita.
La rotazione di 5° è soltanto empirica e può essere facilmente calibrata per adattarsi a tracciati particolarmente curvi. Il programma può anche essere migliorato in modo che il robot trovi da solo l'inizio della riga nera e poi prosegua lungo il tracciato.
Nella prossima sezione di LABORATORIO inizia il progetto di un braccio meccanico.

I programmi "Inseguitore"

MICRO BBC

```
10 REM **** INSEGUITORE DI LINEA - BBC ****
20:
30 PROCinizializza
40 PROCsegui_linea
50 END
60:
70 DEF PROCsegui_linea
80 REPEAT
90 PROCmovimento(avanti,1)
100 PROCverifica_fr
110 UNTIL flagfine=1
120 FINEPROC
130:
140 DEF PROCverifica_fr
150 valfr=?REGDAT AND 48
160 IF valfr=32 THEN PROCgira(destra,5)
170 IF valfr=16 THEN PROCgira(sinistra,5)
180 IF valfr=0 THEN flagfine=1
190 FINEPROC
200:
210 DEF PROCinizializza
220 DDR=&FE62:REGDAT=&FE60:?DDR=15
230 ?REGDAT=1
240 avanti=4:indietro=2:sinistra=6:destra=0
250 rapp_pd=3.34446:rapp_pa=379/90
260 flagfine=0
270 FINEPROC
280:
290 DEF PROCmovimento(dir,distanza)
300 ?REGDAT=(?REGDAT AND 1)OR dir
310 impulsi=rapp_pd*ABS(distanza)
320 FOR I=1 TO impulsi:PROCimpulso:NEXT I
330 FINEPROC
340:
350 DEF PROCgira(dir,angolo)
360 ?REGDAT=(?REGDAT AND 1)OR DIR
370 impulsi=rapp_pa*angolo
380 FOR I=1 TO impulsi:PROCimpulso:NEXT I
390 FINEPROC
400:
410 DEF PROCimpulso
420 ?REGDAT=(?REGDAT OR 8)
430 ?REGDAT=(?REGDAT AND 247)
440 FINEPROC
```

COMMODORE 64

```
10 REM **** INSEGUITORE DI LINEA - CBM ****
20:
30 GOSUB 2000:REM INIZIALIZZAZIONE
40 GOSUB 1000:REM SEGUI LINEA
50 END
60:
1000 REM **** SEGUI LINEA ****
1010 DR=AV:DS=1:GOSUB 2500:REM MUOVE
1020 GOSUB 1500:REM VERIFICA FR
1030 IF EF<>1 THEN 1000
1040 RETURN
1050:
1500 REM **** VERIFICA FR ****
1510 LV=PEEK(REGDAT)AND 48
1520 IF LV=32 THEN DR=DE:DS=5:GOSUB 3000:REM GIRA
1530 IF LV=16 THEN DR=SI:DS=5:GOSUB 3000:REM GIRA
1540 IF LV=0 THEN EF=1
1550 RETURN
1560:
2000 REM **** INIZIALIZZAZIONE ****
2010 DDR=56579:REGDAT=56577:POKEDDR,15
2020 POKEREGDAT,1
2030 AV=4:ND=2:SI=6:DE=0
2040 PD=3.34446:PA=379/90
2050 EF=0:REM FLAGFINE
2060 RETURN
2070:
2500 REM **** MOVIMENTO (DR,DS) ****
2510 POKE REGDAT,(PEEK(REGDAT)AND 1)OR DR
2520 PL=PD*ABS(DS)
2530 FOR I=1 TO PL:GOSUB 3500:NEXT I
2550 RETURN
2560:
3000 REM **** GIRO (DR,DS) ****
3010 POKE REGDAT,(PEEK(REGDAT)AND 1)OR DR
3020 PL=PA*ABS(DS)
3030 FOR I=1 TO PL:GOSUB 3500:NEXT I
3040 RETURN
3050:
3500 REM **** IMPULSI ****
3510 POKE REGDAT,PEEK(REGDAT)OR 8
3520 POKE REGDAT,PEEK(REGDAT)AND 247
3530 RETURN
```

SPECTRUM

Usare la versione per CBM, cancellando le linee 2010, 2020, 2510 e 3010 e apportando le seguenti modifiche:

```
10 REM **** INSEGUITORE DI LINEA - SPECTRUM ****
15 CLEAR 32499:ST=32500:GOSUB 4500
1510 LET NM=48:GOSUB 4000:LET LV=USR ST
3510 OUT 31,DR+9
3520 OUT 31,DR+1
4000 REM **** ESEGUE AND ****
4010 POKE ST+1,IN 31
4020 POKE ST+3,NM:RETURN
4500 **** CARICATORE IN CODICE MACCHINA ****
4510 FOR I=ST TO ST+8
4520 READ A:POKE I,A
4530 NEXT I
4540 DATA 62,0,14,0,161,6,0,79,201
4550 RETURN
```

Viaggio nella didattica



Anche i giochi di simulazione e di avventura possono essere utilizzati per insegnare

I giochi su computer sono divertenti e questo tutti lo sanno; ma possono essere anche istruttivi e utilizzabili nell'ambito dell'insegnamento scolastico? A quanto pare sì. Molti insegnanti concordano sul fatto che anche l'insegnamento deve essere in parte ricreativo, pertanto l'impiego di giochi nel processo apprenditivo ha già preso piede in diversi paesi. La Gran Bretagna è stata forse la prima nazione tra quelle europee a interessarsi attivamente in tal senso, sperimentando i microcomputer nelle scuole fin dagli inizi. Contemporaneamente, si è assistito ad un nuovo sviluppo della "teoria dei giochi" in psicologia. Il computer, anche in questo campo, può dunque essere utilizzato con grande profitto.

I due generi di gioco predominanti su computer, commercialmente parlando, sono stati fino ad ora quello del tipo "space invaders" e quello delle simulazioni/avventure. I più recenti sviluppi software, comunque, rendono la distinzione tra i due sempre meno netta, anche se spesso la loro ispirazione è riconoscibile.

I giochi del primo genere, nonostante la loro enorme popolarità, hanno influenzato ben poco la produzione di software educativo. Questo argomento è stato anche ripreso in un libro (*Mind at Play, The Psychology of Video Games*), in cui gli autori, Elizabeth e Geoffrey Loftus, riportano i commenti in merito di alcuni esperti psicologi . L'unico fattore interessante dei giochi di tipo space invaders, sotto un profilo educativo e terapeutico, è che la prontezza di riflessi e il coordina-

Bravi al gioco

I giochi di simulazione e di avventura favoriscono l'esplorazione, da parte degli allievi, di situazioni o esperimenti altrimenti impossibili da eseguire realmente, a causa del loro costo oppure della loro potenziale pericolosità. Ben lungi dal costituire una perdita di tempo, molti giochi si dimostrano invece ottimi sussidi didattici nel processo di apprendimento



per concessione de SIMULATION DATABASE PROJET, Glasgow College of E



IAN McKINNELL

Viaggio nel tempo

Il "Simulation-Database Project", sviluppato presso il St. Andrews College di Glasgow, impiega il computer non come sostituto, bensì come complemento nelle attività didattiche tradizionali. Il package è composto da tre elementi: un gioco di avventura, una raccolta di testi memorizzati su file (contenenti informazioni più approfondite circa le situazioni del gioco) e un database in cui sono raccolti dati bibliografici di vario genere. La classe di studenti viene suddivisa in gruppi, ciascuno dei quali, a turno, assume il ruolo di un personaggio del gioco. Da questa esperienza possono scaturire diverse attività secondarie da svolgere sia in classe che fuori

mento dei movimenti necessari possono in certi casi stimolare soggetti colpiti da disordini mentali o da traumi cerebrali.
Alcuni programmi orientati alla didattica infantile hanno tentato di riproporre argomenti tradizionali sotto forma di tali giochi. È il caso di *Spell Invaders* e *Maths Invader*, in cui il bambino vince se "abbatte" gli alieni rispondendo correttamente alle domande poste dal programma. Tuttavia, simili tentativi hanno riscosso tiepidi successi.
I giochi di simulazione e quelli di avventura, al contrario, sembrano avere potenzialità notevolmente superiori. Le simulazioni consentono di ricreare condizioni ambientali e sperimentali altrimenti inaccessibili agli studenti (la camera di combustione di un motore, una centrale atomica, ecc.). Le avventure, con i loro scenari spesso fantastici, ma talvolta anche realistici, stimolano al ragionamento e alla deduzione. Questo genere di programmi viene già utilizzato in corsi di tirocinio industriale e anche in qualche scuola.
Per esempio il programma *Ballooning* (scritto da Hill McGibbon per lo Spectrum e il Commodore 64) è un tipico gioco di simulazione in cui i bambini sperimentano, senza i rischi del caso, il volo su mongolfiera. Pur trattandosi di un gioco, gli elementi della simulazione rispettano le leggi della fisica e i partecipanti, oltre a manovrare i relativi comandi, si devono destreggiare tra altimetri, indicatori di carburante e di pressione e quadranti di discesa/ascesa. Il programma prevede anche stazioni di rifornimento a terra, alle quali occorre avvicinarsi con estrema perizia. Programmi di questo genere hanno l'evidente pregio di mantenere attivo, grazie al loro aspetto ludico, l'interesse dell'allievo abituandolo al tempo stesso a recepire nozioni che possono in seguito essere sviluppate e approfondite dall'insegnante.

Vantaggi per l'istruzione

Oltre ad essere poco costosi e privi di pericoli, i giochi di simulazione offrono uno strumento didattico di notevole portata. I bambini possono concentrarsi sulle cause e gli effetti di particolari esperimenti, che sono sempre riproducibili in qualsiasi momento e, spesso, anche senza il diretto intervento dell'insegnante. La rapidità con cui si svolge in natura un evento fisico non rappresenta più un ostacolo, poiché molti programmi consentono di "congelare" le varie fasi dell'esperimento, per soffermarsi su qualche particolare importante. Di contro, esperimenti fisici che nella realtà richiederebbero giorni o intere settimane possono essere riprodotti con sufficiente approssimazione nel breve arco di qualche minuto. Un modello computerizzato della realtà semplifica sistemi troppo complessi per essere spiegati nella loro globalità. Inoltre, poiché la simulazione può essere ripetuta anche mutando le condizioni di partenza, il numero di esperimenti possibili è pressoché infinito. Lo studente, che viene messo in grado di controllare molte, se non tutte, le variabili di un esperimento (ivi compreso il tempo), riesce dunque a formarsi una visione più completa del fenomeno osservato. Ovviamente esistono alcune limitazioni, nelle simulazioni, ed è bene che l'insegnante e lo studente ne siano al corrente. Quasi sempre,

infatti, per la limitata quantità di memoria a disposizione, il modello computerizzato è fortemente semplificato in alcune sue parti, rispetto alla situazione reale. Inoltre, gli esperimenti mancano spesso di quei fattori imponderabili tipici dell'esperienza vera. Pertanto, tutti i programmi di simulazione, per quanto efficaci e sofisticati siano, devono essere usati in un contesto più globale e corroborati da ampie spiegazioni sul loro funzionamento e sui loro limiti.
Un fatto è certo: una simulazione, per quanto buona, non è mai la realtà: gli studenti (ma anche gli insegnanti) non devono dunque essere tratti in inganno dall'apparente realismo di un programma.
In sostanza, le simulazioni devono supplire all'impossibilità di compiere, per il loro costo o per la loro pericolosità, determinati esperimenti pratici. È assurdo che invece si occupino di campi che possono benissimo fare a meno di programmi di simulazione. Tanto per fare un esempio, esiste un programma scritto per l'Apple, che consente di progettare semplici circuiti elettrici composti da lampadine e batterie: c'è da chiedersi come mai chi ha il denaro per acquistare computer e programma non possa permettersi qualche lampadina e un paio di pile vere!

Creazione di micromondi

La maggior parte della gente, sia adulti che bambini, ha una notevole tendenza a fantasticare. Negli ultimi anni si è assistito al fiorire della letteratura fantastica (*L'Hobbit*; *2001: Odissea nello spazio*, ecc.). Di pari passo, i giochi di tipo fantastico/avventuroso, in cui il giocatore assume il ruolo di protagonista hanno ottenuto un successo sempre maggiore. Ciò ha coinciso con una presa di coscienza generale sull'importanza del gioco "attivo" con finalità terapeutiche e didattiche.
La potenzialità di questi giochi è enorme e la possibilità di creare su computer delle specie di "micromondi", magari arricchiti da una buona grafica e dedicati a particolari ambienti geografici o periodi storici, ha già stimolato più di un produttore di software.
Ad esempio, quale miglior metodo per insegnare la storia ad un bambino, di quello che consente di calarsi in un'avventura ambientata in una particolare periodo storico? Per sopravvivere, nell'avventura, gli sono necessarie informazioni su tale periodo, che può estrarre da un'apposita banca dati, abbinata al gioco (e magari utilizzabile anche in modo autonomo). Il progetto sembra ambizioso, ma in realtà si tratta semplicemente di conciliare, secondo il vecchio detto, "l'utile al dilettevole": rendere piacevole l'apprendimento di una materia, sfruttando i mezzi che la tecnologia di oggi mette a disposizione.
Un progetto simile, chiamato *Database per simulazioni* è stato realizzato presso il St. Andrews College di Glasgow, con lo scopo di indagare come un computer possa diventare un efficace strumento pedagogico. La struttura della simulazione è basata su di un gioco d'avventura: il bambino si trova calato in un micromondo storico, in cui accadono eventi o si presentano situazioni che richiedono una partecipazione attiva e ragionata. A ciascun ambiente del micromondo è abbinato un testo, richiamabile all'occorrenza, in cui sono riportate notizie storiche e informazioni pertinenti. Un secondo database, accessibile autonomamente, ma con espliciti riferimenti ai "luoghi" del gioco, fornisce ulteriori informazioni e una ricca bibliografia (libri, film e altri "media") sugli argomenti toccati durante il gioco.
È curioso notare come, nel progetto sviluppato al St. Andrews College, il computer svolga un ruolo importante, ma non predominante: il suo uso è volutamente ridotto al minimo indispensabile, facendo parte di un'attività didattica più ampia.
Altre esperienze simili, anche se in scala ridotta, hanno dimostrato che questi metodi possono condurre a risultati pienamente soddisfacenti. Gli studenti, infatti, traggono spunto da quanto appreso tramite l'impiego del computer per avviare tutta una serie di attività collaterali o complementari: interviste, saggi e perfino la stesura di testi teatrali o di articoli giornalistici.

L'UOMO E IL GIOCO

Alcuni educatori e molti genitori impallidiscono al pensiero di studenti che trascorrono parte del loro tempo "giocando", anziché immersi nello "studio": vedere un gruppo di bambini impegnati a esplorare il funzionamento della tartaruga e del LOGO può, come prima impressione superficiale, far credere che si tratti di un gioco e nulla più. Al contrario, le numerose prove fatte a tal riguardo hanno dimostrato che simili esperienze hanno valore educativo.
Altri genitori e insegnanti, all'estremo opposto affermano che l'apprendimento avviene seguendo un solo percorso: il gioco. Questa tesi non è poi tanto strampalata, anzi è sostenuta non solo dalle più recenti teorie sull'apprendimento, ma anche dalla filosofia e dalla storia. Il professor J. Huizinga, in un libro pubblicato nel 1944 quando egli era docente presso l'Università di Layden in Germania, sostiene che il gioco non solo è un elemento indispensabile nel processo apprenditivo, ma che esso svolge inoltre un ruolo importantissimo nella stessa formazione culturale della società. Il libro, intitolato "Homo Ludens" (L'uomo che gioca), fissa alcuni punti fondamentali sull'importanza del gioco:

- Il gioco ha una funzione molto significativa, in quanto spesso riproduce, in piccolo, situazioni reali in cui il bambino (o l'uomo) esplica se stesso
- Il gioco non è affatto una perdita di tempo, nelle occasioni e misure adeguate
- Il gioco è spontaneo e istintivo. I bambini giocano, più o meno coscientemente, con lo scopo di esplorare e apprendere le caratteristiche dell'ambiente circostante
- La civiltà contemporanea soffre di un'eccessiva schematizzazione dei giochi: quando un gioco assume una struttura troppo complessa, esso perde le caratteristiche di un vero gioco



Il computer... agricolo

Il package di programmi Farmfax aiuta nella gestione delle attività agricole



I computer vengono oggi usati nei più disparati settori della società; non deve quindi meravigliare la comparsa sul mercato di una serie di pacchetti software dedicati agli agricoltori. Desta invece sorpresa l'apparecchio al quale sono destinati molti di questi programmi: il Dragon. Questo computer, oggi fabbricato in Spagna, sta riscuotendo un crescente successo tra gli agricoltori di molti paesi europei.

Uno di questi pacchetti "agricoli" è il *Farmfax*, ideato da Geoffrey Paterson, egli stesso un agricoltore. Il fatto che ciascun pacchetto sia dedicato ad una particolare attività agricola sottolinea la notevole competenza delle persone che hanno progettato e codificato (in linguaggio Assembly si badi bene!) i diversi programmi. Grazie ad una struttura abbastanza semplice, i programmi risultano comprensibili anche ai meno esperti di computer.

I diversi programmi sono registrati su cartucce ROM ed entrano in funzione non appena l'utente innesta la cartuccia nel computer. Alcune cartucce possiedono anche una RAM aggiuntiva, che semplifica e velocizza l'archiviazione delle informazioni, anche se è sempre consigliabile tenere copie di riserva su nastro.

L'introduzione dell'informatica in agricoltura ha già dato risultati di grande interesse. Un agricoltore inglese, Robin Dunkerley, proprietario di 500 acri di terreno, per metà coltivati e per metà a pascolo, ha pianificato con successo la propria produzione di carne e di latte con il pacchetto *Farmfax*, riuscendo con esso a determinare una specie di "Indice delle nascite", (il numero di giorni durante i quali una mucca produce latte non essendo gravida). Questo indice è importante, poiché incide notevolmente sulla produzione annuale di latte e consente di programmare la frequenza degli accoppiamenti per ciascun capo di bestiame.

Schede individuali per le mucche

Per meglio comprendere il funzionamento del package *Farmfax*, esaminiamo il programma *Individual Cow Records*, che permette di archiviare (su nastro nella versione più economica, altrimenti su cartuccia) le informazioni relative a ciascun capo di bestiame, fino a un massimo di 240. Detto per inciso, il funzionamento delle due versioni (nastro o cartuccia) è pressoché identico: i dati vengono comunque caricati ed elaborati in memoria prima di essere salvati. Sebbene molti considerino il nastro uno strumento d'archiviazione eccessivamente lento, in questo caso rappresenta un accettabile compromesso tra costo e prestazioni. Ciò che interessa all'agricoltore, più della velocità, è potersi esimere dall'eseguire a mano i calcoli necessari per decidere quando vendere una mucca la cui produzione di latte sta calando, confrontando le spese in foraggi per il suo mantenimento e le entrate relative alla vendita del suo latte. Per di più una pianificazione del lavoro sul proprio computer (anche se a nastro), invece di doversi affidare ad un servizio esterno di elaborazione a pagamento, risulta a lungo andare molto più semplice e conveniente.

Prima di acquistare un computer, Dunkerley si serviva dell'elaboratore di una società di raccolta del latte alla quale versava una

tariffa mensile. Le informazioni relative alle diverse mucche venivano immesse nel computer alla fine di ogni mese e trascorrevano oltre tre settimane prima poter ricevere un bilancio completo del mese precedente. Con questi ritmi, le adeguate modifiche da apportare nella produzione del latte, in base alle informazioni ricevute, avvenivano con forti ritardi. Quando finalmente si decise all'acquisto, Dunkerley optò per un sistema composto da un Dragon, da una stampante e dal software *Farmfax*, il cui costo complessivo, relativamente basso, venne rapidamente ammortizzato. Oggi egli può elaborare senza alcuna spesa e in pochi minuti le informazioni di cui ha bisogno.
Alla fine di ogni mese, Dunkerley dedica circa un'ora al riordino e all'elaborazione del proprio sistema d'informazioni. Per primo egli esegue il programma per la gestione della produzione di latte (*Dairy Management Program*) che calcola il bilancio economico di questa attività, il margine di guadagno relativo alle vendite del latte e alle spese sostenute per il foraggio, le entrate lorde, la quantità di foraggio consumata per produrre un litro di latte e la produzione totale di latte, inaieme ad altre importanti informazioni. Poi Dunkerley lancia il programma *Dairy Ration*, che verifica se la quantità di latte prodotto corrisponde alle previsioni e analizza la resa del foraggio. Comunicando al computer i dati relativi alla quantità di megajoule di energia, alle proteine crude, alle sostanze acquose presenti in ogni chilo di foraggio, al prezzo del foraggio e alla eventuale produzione in proprio di alimenti per il bestiame, Dunkerley può calcolare i costi totali di produzione e i costi giornalieri per ogni capo.

Funzionamento del Farmfax

Immessi i dati relativi alla produzione giornaliera di latte di ogni mucca, il programma *Individual Cow Records* fornisce un'accurata registrazione alla fine di ogni mese. Se uno dei parametri del sistema viene modificato, il programma ricalcola automaticamente tutte le informazioni precedentemente immesse, riducendo di circa un terzo i tempi necessari alla gestione dell'azienda. Inoltre aggiorna costantemente l'utente sulla sua situazione finanziaria, comunicandogli eventuali scoperti bancari, ed esegue un accurato bilancio preventivo.
I diversi moduli del programma *Farmfax* funzionano in modo semplice e comprensibile, iniziano con un sintetico menu e sono corredati da manuali tascabili. Il loro uso si apprende in pochi minuti.
Il programma *Cullcow*, che aiuta l'agricoltore a decidere se vendere o meno le mucche la cui produzione di latte cala al di sotto di un certo livello, inizia con un menu di due opzioni, del tipo:

PAGINA 1 — INFORMAZIONI SUL PREZZO
PAGINA 2 — MEGLIO COMPRARE/VENDERE

Se sceglie la pagina 1, l'agricoltore deve inserire il mese e l'anno correnti, quindi i tassi d'interesse bancario applicabili, il prezzo del latte, la produzione giornaliera in litri di ciascuna mucca e i costi mensili. Per risparmiare tempo, le informazioni inserite vengono automaticamente passate alle altre procedure del programma. Tali dati, comunque, possono essere corretti manualmente se, ad esempio, i tassi d'interesse variano nei mesi successivi.
Se viene selezionata la pagina 2, il programma chiede all'agricoltore se vuole acquistare o vendere le mucche, il numero di capi da trattare, il prezzo previsto e il mese di vendita. Quindi il computer elabora le diverse informazioni e pone un asterisco accanto alla mucca che conviene mettere in vendita. In qualsiasi momento, premendo il tasto BREAK, l'utente può tornare all'indice per correggere i dati introdotti, se necessario.
Gli altri pacchetti funzionano in modo simile, ma i programmi più sofisticati presentano menu d'apertura più complessi. La prima pagina del *Dairy Ration Formulation*, ad esempio, visualizza un menu del tipo:

PAGINA 1 — TABELLA DEGLI ALIMENTI
2 — MINERALI
3 — ANALISI DEI DATI
4 — PARAMETRI DI PRODUZIONE
5 — DOSAGGIO
6 — RAZIONI DI FORAGGIO
7 — ARCHIVIAZIONE/RICHIAMO

Se la cartuccia impiegata è dotata di memoria RAM, il programma visualizza il menu; in caso contrario, salta automaticamente alla pagina 7, permettendo così di richiamare dal nastro le informazioni registrate in precedenza. I dati inseriti a pagina 1, relativi agli alimenti e ai loro costi, vengono copiati automaticamente sulla pagina 2 nella quale l'utente registra i minerali contenuti in ciascun alimento.
A pagina 3 viene analizzata la quantità di sostanze energetiche, elementi acquosi e proteine contenute nel foraggio per unità di costo. Inserendo a pagina 4 le previsioni di produzione in termini di peso corporeo della mucca, giorni di gravidanza, produzione di latte in litri e di burro, è possibile stabilire la razione giornaliera di alimenti e calcolare immediatamente i costi totali del foraggio per un dato numero di giorni in base al numero dei capi.
Ovviamente i pacchetti per l'agricoltura sono molto specializzati: sono scritti da esperti del settore e dedicati ad altri esperti. Il loro obiettivo è quello di consentire agli agricoltori di integrare fra loro le diverse attività agricole e di gestire l'azienda con criteri industriali.

Farmfax suite: Per il Dragon (in lingua inglese)
Distributori: Farmfax Computer Systems (GB)
Formato: Cartuccia

Alimenti per la ciurma

Lo sviluppo del gioco prosegue con le ordinazioni delle provviste

La giusta quantità di scorte

Il giocatore può scegliere tra quattro tipi di provviste: verdure, frutta, carne ed acqua. I nomi di questi prodotti sono contenuti nella matrice P$( ) e le unità di misura corrispondenti in U$( ). Per decidere la quantità di scorte da acquistare, il giocatore deve basarsi sul prezzo unitario e sulle razioni settimanali necessarie per ciascun membro dell'equipaggio. Le matrici che contengono queste informazioni sono PC( ) e PN( ). Le quattro matrici sono dimensionate all'inizio del programma. In una quinta matrice, PA( ), viene registrata la quantità di ogni ordinazione

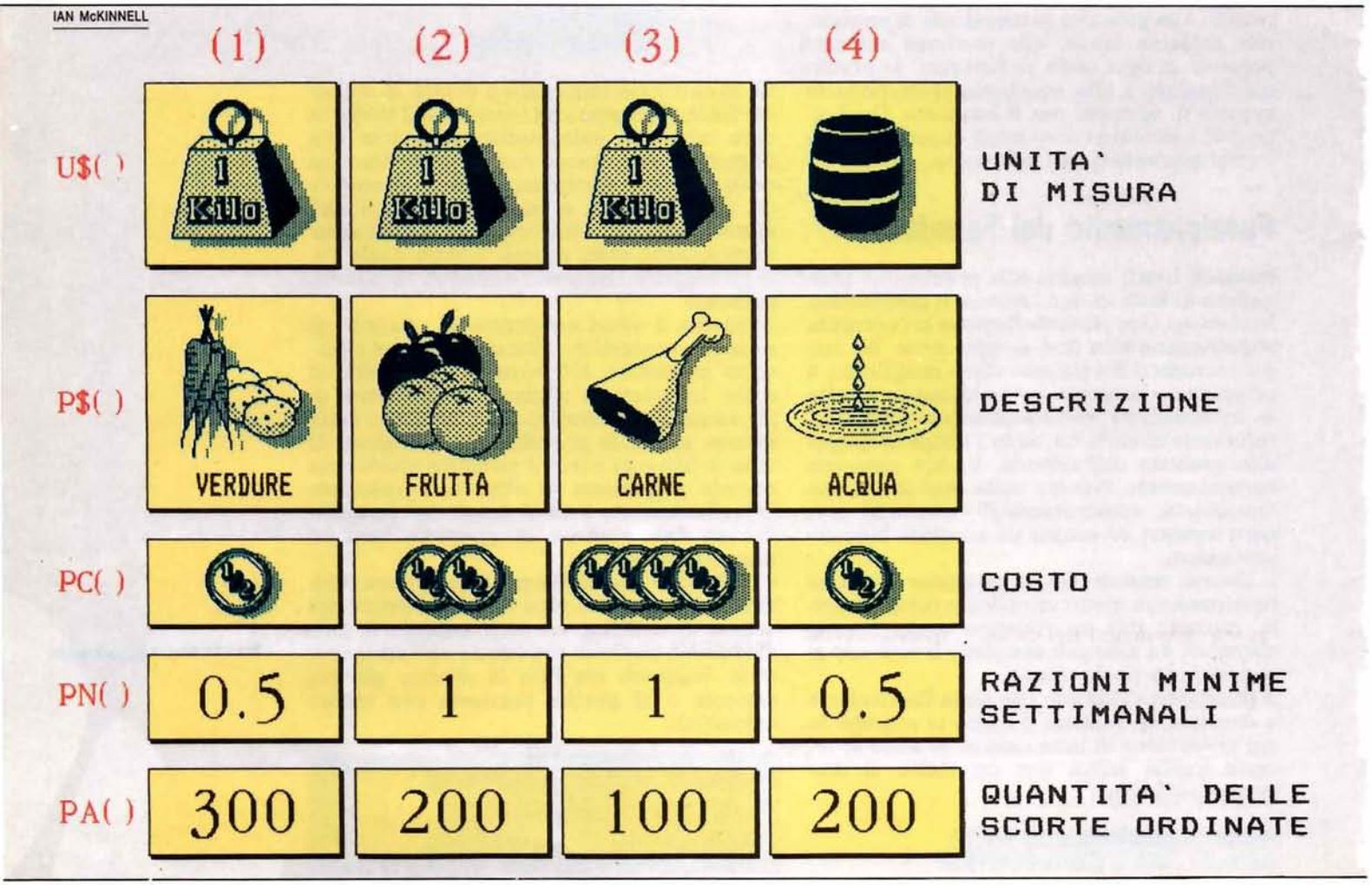
Nella prima fase del gioco di simulazione *Nuovo Mondo*, il giocatore assolda l'equipaggio. Al termine di questa operazione sullo schermo compare il messaggio PREMI QUALSIASI TASTO. La seconda fase del gioco è invece dedicata all'acquisto delle scorte. Il giocatore sa che il viaggio dura circa 8 settimane, quindi deve caricare una quantità di cibo e d'acqua sufficiente.

Il programma calcola le scorte necessarie per 8 settimane e verifica che vengano acquistate provviste sufficienti. In caso contrario, avverte che l'equipaggio rischia di patire la fame o la sete e consente di modificare la quantità delle scorte; ciò avviene anche nel caso in cui il giocatore non possegga denaro a sufficienza per pagarle: il programma visualizza anche la situazione finanziaria relativa alla procedura d'acquisto delle scorte. Come per l'assunzione della ciurma anche in questo caso è necessario definire una serie di matrici.

```
19 **** MATRICI DI APPROVVIGIONAMENTO ****
20 DIM U$(4): U$(1) = "KG.": U$(2) = "KG.": U$(3) = "KG.": U$(4) = "BARILI"
21 DIM P$(4): P$(1) = "ORTAGGI": P$(2) = "FRUTTA": P$(3) = "CARNE":
   P$(4) = "ACQUA"
22 DIM PA(4)
23 DIM PC(4):PC(1) = .5:PC(2) = 1:PC(3) = 2:PC(4) = .5
24 DIM PN(4):PN(1) = 2:PN(2) = 1:PN(3) = 1:PN(4) = .5
```

La verdura, la frutta e la carne vengono pesati in chili (KG.), l'acqua in barili. La linea 20 definisce una matrice U$ a quattro elementi per le diverse unità di misura. La registrazione di questi parametri su matrice consente di risparmiare memoria rispetto all'impiego delle procedure di visualizzazione. Inoltre la matrice può essere ampliata con altre unità di misura (ad esempio "sacchi" di farina).

La linea 21 definisce una seconda matrice P$( ), anch'essa a quattro elementi, che contiene i nomi dei quattro tipi di provviste. Le quantità delle scorte acquistate vengono depositate in una terza matrice, PA( ). Un chilo di ortaggi costa mezza moneta d'oro, un chilo di frutta



una moneta, un chilo di carne due monete ed un barile d'acqua mezza moneta. Questi valori vengono registrati, alla linea 23, nella matrice PN(), che può essere facilmente estesa per provviste d'altro tipo. La sequenza di acquisto delle scorte non segue l'ordinamento dei parametri all'interno della matrice: viene definita automaticamente dal programma.
La matrice PN() alla linea 24 contiene la quantità di provviste necessarie a ciascun membro dell'equipaggio per mantenersi in salute. Poiché ogni uomo consuma, ad esempio, due chili di ortaggi alla settimana, al primo elemento della matrice, PN(1), viene assegnato il valore 2.
La linea 500 del primo modulo presentato trasferisce il controllo del programma alla linea 1000, per l'assunzione dell'equipaggio. Al termine il controllo torna alla linea 500. Per chiamare la routine di acquisto delle scorte occorre inserire la seguente chiamata GOSUB dopo la linea 500:

550 GOSUB 2000: REM APPROVVIGIONAMENTO

Per prima cosa la routine visualizza alcune informazioni sull'acquisto delle scorte. La variabile CN, definita nel modulo precedente, viene usata alla linea 2040 per ricordare al giocatore il numero dei membri dell'equipaggio.
La procedura di approvvigionamento è contenuta in un ciclo iterativo che si ripete quattro volte, una per ogni tipo di provvista, e il contatore di ciclo T (linea 2050) viene incrementato di uno a ogni passaggio. Inoltre la routine visualizza le razioni di cibo settimanali per ciascun membro dell'equipaggio, calcolate in base ai parametri contenuti nella matrice PN(). Come si è visto, le unità di misura delle diverse provviste sono registrate nella matrice U$(). La linea 2085 visualizza il nome della provvista contenuto nella matrice P$(). La linea 2088 provvede a visualizzare il costo dell'acqua, modificando BARILI in BARILE.
Quindi il programma chiede al giocatore la quantità di ciascuna ordinazione (linee 2100 e 2110). Il valore digitato viene visualizzato dalla linea 2130 e caricato nel corrispondente elemento della matrice P$() alla linea 2140. Se l'ordinazione è zero, la linea 2140 avverte dei possibili disagi che ne conseguono, altrimenti il programma calcola, con la formula CN*8*PN(T), se la quantità ordinata e caricata nella matrice PA() è sufficiente per un viaggio di 8 settimane (linea 2170). CN rappresenta il numero dei membri dell'equipaggio e PN(T) la razione settimanale di quel particolare alimento. Se le scorte sono scarse, un messaggio avvisa il giocatore che l'equipaggio rischia di patire la FAME (o la SETE, verificando se si tratta del quarto passaggio nel ciclo). A questo avviso segue un invito a modificare l'ordinazione. Se il giocatore, pur essendo a corto di denaro, risponde affermativamente, il controllo passa alla linea 2400. Se la nuova quantità ordinata è diversa dalla precedente, il programma azzera l'elemento corrispondente della matrice PA() e, dopo aver sottratto 1 dal contatore di ciclo T, ripete la procedura di acquisto.
Se il giocatore conferma l'ordine precedente, il programma visualizza la quantità di denaro in suo possesso e una lista delle provviste ordinate. Può accadere che il giocatore acquisti scorte sufficienti per l'intero viaggio ma non possieda il denaro per pagarle (linea 2260): in tal caso il programma avverte il giocatore che il denaro non basta per pagare le scorte (linee 2270-2290) e lo invita a inserire uno nuovo ordine. Se gli acquisti vengono confermati, il loro valore viene sottratto dal denaro rimasto, contenuto in MO.
La linea 2410, che fa parte di un altro ciclo iterativo, visualizza la lista delle scorte via via acquistate, usando le variabili U$, PA() e P$(). L'ammontare del denaro residuo è visualizzato dalla linea 2480. Il ciclo termina alla linea 2490 e, infine, il giocatore passa alla fase di gioco successiva battendo un tasto in risposta al solito messaggio PREMI UN TASTO QUALSIASI, anche se, per adesso, ciò conclude il gioco.

MODULO 2

```
2000 REM ••• APPROVVIGIONAMENTO •••
2005 PRINT CHR$(147):REM PULIZIA SCHERMO
2010 S$ = "FASE 2 - APPROVVIGIONAMENTO*"
2015 GOSUB 9100
2020 S$ = "------- -------------------------*"
2025 GOSUB 9100
2030 GOSUB 9200:PRINT
2040 PRINT "HAI INGAGGIATO ";CN;" UOMINI."
2045 GOSUB 9200:GOSUB 9200
2050 FOR T = 1 TO 4
2055 PRINT
2060 PRINT "PER OGNI SETTIMANA DI VIAGGIO "
2070 PRINT "CIASCUN MEMBRO DELL'EQUIPAGGIO "
2075 PRINT "HA BISOGNO DI ALMENO ";PN(T);" ";U$(T);
2085 PRINT "DI ";P$(T)
2086 PRINT "A ";PC(T);" MONETE D'ORO AL ";
2088 IF T < > 4 THEN PRINT U$(T);:GOTO 2092
2090 PRINT LEFT$(U$(T),LEN(U$(T))-1);"E";
2092 PRINT ")"
2095 GOSUB 9200:PRINT:GOSUB 9200
2100 PRINT "QUANTI ";U$(T);" DI ";P$(T)
2110 S$ = "VUOI ACQUISTARE*":GOSUB 9100
2120 PRINT
2130 INPUT P$
2140 PA(T) = VAL(P$):GOSUB 9200
2150 IF PA(T) > ((CN*8*PN(T))-1) THEN 2260
2160 IF PA(T) = 0 THEN PRINT "SE NON ACQUISTI ":GOTO 2180
2170 PRINT "SE COMPERI SOLO ";PA(T);U$(T);" DI "
2180 PRINT P$(T);",":GOSUB 9200
2190 PRINT " QUALCUNO PATIRA'";
2200 S$ = "LA FAME"
2210 IF T = 4 THEN S$ = "LA SETE"
2220 PRINT S$;"!":GOSUB 9200
2230 S$ = "MODIFICHI L'ORDINAZIONE (S/N)":GOSUB 9100
2240 INPUT P$:P$ = LEFT$(P$,1)
2242 IF P$ < > "S" AND P$ < > "N" THEN 2230
2245 IF P$ = "N" THEN 2400
2250 PA(T) = 0:T = T-1:GOTO 2410
2260 IF PA(T)*PC(T) > MO THEN 2270
2265 GOTO 2400
2270 S$ = "NON HAI ABBASTANZA DENARO PER":GOSUB 9100
2280 PRINT PA(T);" ";
2290 PRINT U$(T);" DI ";P$(T):GOSUB 9200
2300 S$ = "RIPROVA!* ":GOSUB 9100:PA(T) = 0:T = T-1:GOTO 2410
2400 MO = MO -(PA(T)*PC(T))
2410 PRINT:S$ = "PROVVISTE ATTUALI:* ":GOSUB 9100
2412 GOSUB 9200
2415 FOR TT = 1 TO 4
2420 PRINT PA(TT);U$(TT);" DI ";
2440 PRINT P$(TT)
2450 GOSUB 9200
2460 NEXT
2480 PRINT "MONETE D'ORO RIMANENTI = ";MO
2485 GOSUB 9200:GOSUB 9200
2490 NEXT T
2500 GOSUB 9200:PRINT:S$ = "FINE APPROVVIGIONAMENTO.* ":GOSUB 9100:
     GOSUB 9200
2510 PRINT: S$ = K$:GOSUB 9100:PRINT: GOSUB 9200
2520 GET P$:IF P$ = "" THEN 2520
2999 RETURN
```

Procedura di acquisto

Questa subroutine gestisce l'acquisto delle scorte per il viaggio e deve essere aggiunta al primo modulo, riportato a pag. 1033. Il ciclo principale viene ripetuto 4 volte: ad ogni passaggio il giocatore ordina un diverso tipo di provvista

Basic & Basic

Spectrum:

Inserire le seguenti modifiche:

2005 CLS

2240 INPUT P$: LET P$ = P$(1 TO 1)

Micro BBC:

Inserire la seguente modifica:

2005 CLS

DATABASE

Ecco la penultima tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi

DESCRIZIONE	INDIRIZZO
Set FAC#1=1 pone FAC#1=1	**A699**
Set FAC#1=1/FAC#1 pone FAC#1=1	**A6A5**
Set FAC#1=A pone FAC#1=A	**A2ED**
Set FAC#1=FAC#1*(&4B) pone FAC#1=FAC#1*(&4B)	**A606**
Set FAC#1=FAC#1*(&4B);chk ov/flow pone FAC#1=FAC#1*(&4B); controlla overflow	**A656**
Set FAC#1=FAC#1*10 pone FAC#1=FAC#1*10	**A1F4**
Set FAC#1=FAC#1*FAC#2 pone FAC#1=FAC#1*FAC#2	**A613**
Set FAC#1=FAC#1+FAC#2 pone FAC#1=FAC#1+FAC#2	**A50B**
Set FAC#1=FAC#1—(&4B) pone FAC#1=FAC#1—(&4B)	**A4D0**
Set FAC#1=FAC#1—INT(FAC#1) pone FAC#1=FAC#1—INT(FAC#1)	**A486**
Set FAC#1=FAC#1/(&4B) pone FAC#1=FAC#1/(&4B)	**A6E7**
Set FAC#1=FAC#1/FAC#2 pone FAC#1=FAC#1/FAC#2	**A6F1**
Set FAC#1=FAC#1^A pone FAC#1=FAC#1^A	**AB12**
Set FAC#1=FAC#2 pone FAC#1=FAC#2	**A4DC**
Set FAC#1=IAC pone FAC#1=IAC	**A2BE**
Set FAC#2=0 pone FAC#2=0	**A453**
Set FAC#2=FAC#1 pone FAC#2=FAC#1	**A21E**
Set FAC#2=FAC#1*2 pone FAC#2=FAC#1*2	**A23F**
Set IAC=INT(FAC#1) pone IAC=INT(FAC#1)	**A3E4**
Set MAN#1=MAN#1*10 pone MAN#1=MAN#1*10	**A197**
Set MAN#1=MAN#1+A pone MAN#1=MAN#1+A	**A2A4**
Set MAN#1=MAN#1+MAN#2 pone MAN#1=MAN#1+MAN#2	**A178**
Set MAN#1=MAN#2 pone MAN#1=MAN#2	**A4E8**
String addition somma tra stringhe	**9C15**
String assignment assegnazione stringa	**8C1E**
String comparison confronto tra stringhe	**9AE7**

INDIRIZZO	DESCRIZIONE
BBE4	**REPEAT statement routine** routine istruzione REPEAT
BC2D	**Delete a line from program** cancella una linea di programma
BC8D	**Insert a line into program** inserimento di una linea nel programma
BD11	**RUN statement routine** routine istruzione RUN
BD20	**Clear variables** azzera le variabili
BD3A	**Clear stacks** ripulisce gli stack
BD51	**Push FAC#1 onto stack** appoggia FAC#1 sullo stack
BD7E	**Remove FAC#1 from stack** rimuove FAC#1 dallo stack
BD90	**Push anything onto stack** appoggia un qualsiasi elemento sullo stack
BD94	**Push the IAC onto stack** appoggia lo IAC sullo stack
BDB2	**Push a string onto stack** appoggia una stringa sullo stack
BDCB	**Pull a string from stack** estrae una stringa dallo stack
BDDC	**Remove a string from stack** rimuove una stringa dallo stack
BDEA	**Pull the IAC from stack** estrae lo IAC dallo stack
BDFF	**Discard the IAC from stack** elimina IAC dallo stack
BE62	**Load a new program** "Load" di un nuovo programma
BE6F	**Check for a bad program** controllo per "Bad program"
BEC2	**OSCLI statement routine** routine istruzione OSCLI
BEF3	**SAVE command routine** routine comando SAVE
BF24	**LOAD command routine** routine comando LOAD
BF2A	**CHAIN statement routine** routine istruzione CHIAN
BF30	**PTR statement routine** routine istruzione PTR
BF46	**EXT function routine** routine funzione EXT
BF47	**PTR function routine** routine funzione PTR
BF58	**BPUT statement routine** routine istruzione BPUT
BF6F	**BGET statement routine** routine istruzione BGET

L'Electron Plus 3

L'aggiunta di questo dispositivo migliora notevolmente le prestazioni dell'Electron

Una delle principali critiche all'Electron della Acorn riguarda la sua carenza d'interfacce. L'apparecchio adotta il BASIC BBC ad alta velocità, ma il fatto di possedere solo una coppia di prese per il monitor, una porta RF e un'interfaccia per cassette limita notevolmente le sue possibilità d'espansione(manca persino la porta per il joystick) . La Acorn ha più volte annunciato la produzione di moduli d'espansione dotati di interfacce che permettano di portare l'Electron al livello del Micro BBC, ma la loro comparsa sul mercato avviene ad un ritmo molto lento.

La prima espansione prodotta dalla Acorn è stata il Plus 1 (vedere alle pagg. 929-931) che, sebbene possieda numerose interfacce assenti sull'Electron originale, manca tuttavia di un'interfaccia per dischetti che consenta l'elaborazione di grosse quantità di dati.

La situazione è migliorata con il lancio del Plus 3, un sistema a disco completo, costruito appositamente per l'Electron. Questo modulo a forma di "L" si compone di due parti. Il lato più lungo contiene il sistema di archiviazione su dischetto e viene fissato al retro del computer, quello più breve accoglie l'unità a dischetti e si affianca al lato destro dell'Electron, coprendo la presa d'alimentazione. Il Plus 3 possiede un proprio sistema d'alimentazione da 36 watt che fornisce energia sia all'unità a dischetti sia al computer. Come il Plus 1, il Plus 3 si innesta al connettore a pettine posto sul retro dell'Electron e viene fissato da due viti alla base dell'apparecchio per impedire movimenti che potrebbero danneggiare il connettore a pettine. Contrariamente al Plus 1, la nuova unità presenta, sul retro, un ulteriore connettore a pettine sul quale possono essere inseriti lo stesso Plus 1 o futuri moduli d'espansione. Sorprendentemente la Acorn ha dotato il Plus 3 di un'unità a dischetti Sony da 3 1/4 pollici e non del più comune sistema da 5 1/4 pollici. Dietro il drive c'è un altro connettore a pettine per l'innesto di una seconda unità a dischetti. Secondo quanto affermato dalla Acorn, il connettore può accogliere non solo unità da 3 1/4 pollici, ma anche drive da 3 o da 5 1/4 pollici. A questo riguardo, la compagnia ha prodotto un'interfaccia opzionale che permette di collegare al connettore l'unità a dischetti da 5 1/4 pollici del Micro BBC.



CHRIS STEVENS

Assemblaggio modulare

Il modulo Plus 3 della Acorn è progettato per consentire anche l'uso dell'espansione Plus 1. L'introduzione di queste due interfacce ha notevolmente elevato le capacità elaborative dell'Electron, avvicinandole a quelle del Micro BBC

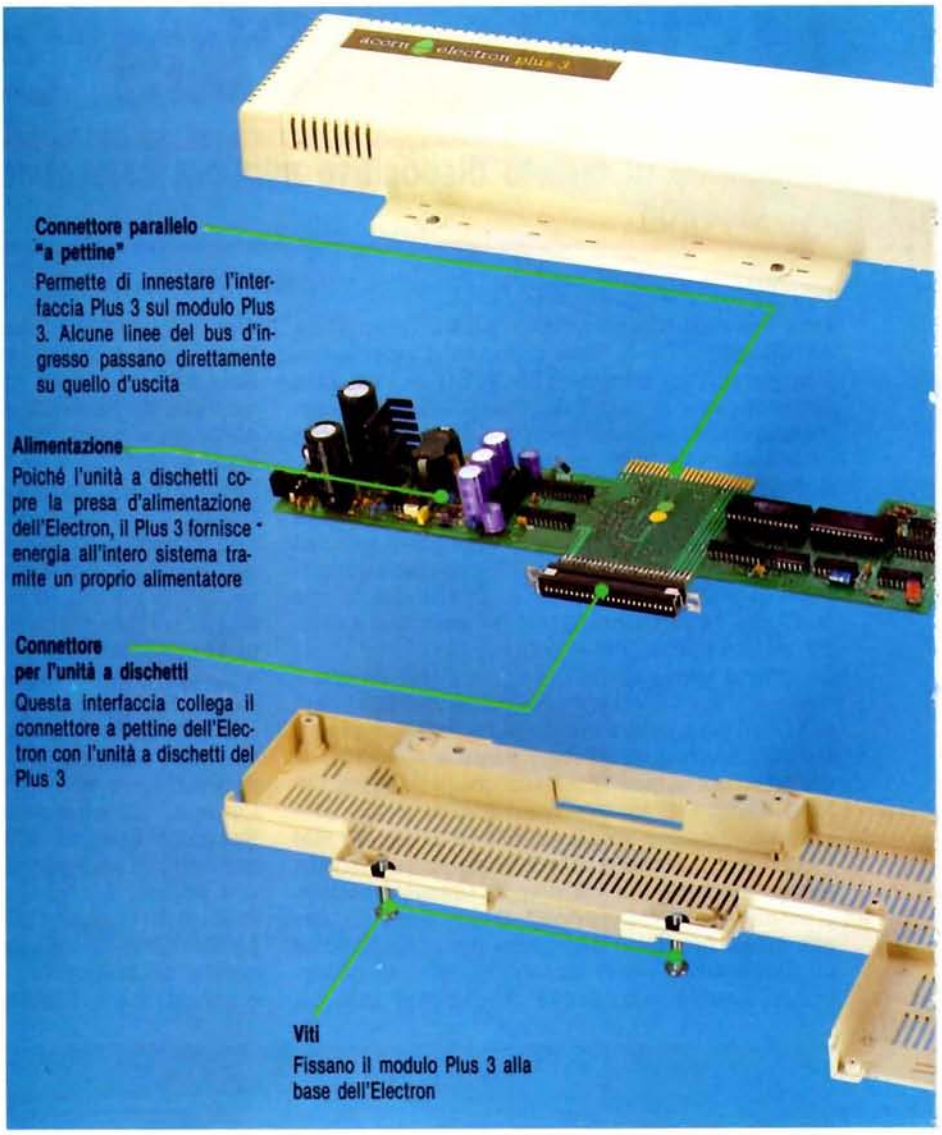
Il Plus 3 è suddiviso internamente in quattro sezioni. Partendo da sinistra, è visibile il circuito di alimentazione (caratterizzato da grossi condensatori), affiancato dai "bus" di collegamento: alcuni di essi raggiungono l'unità a dischetti, mentre altri terminano direttamente sul connettore a pettine per il modulo Plus 1. Seguono i circuiti del sistema di archiviazione su disco (Disk Drive Filing System). I chip di questa sezione contengono una serie di comandi extra, residenti in ROM, che permettono al computer di "dialogare" con l'unità a dischetti e con i chip di gestione del disco. Infine, all'estremità sinistra, c'è l'unità a dischetti a doppia faccia, protetta da una custodia metallica. Quest'ultima svolge una duplice funzione: facilita la dispersione del calore prodotto all'interno del modulo e agisce come gabbia di Faraday, proteggendo l'unità a dischetti da interferenze magnetiche esterne.

I dischetti Sony sono racchiusi in una custodia di plastica rigida e presentano una calottina di protezione sulla finestra di lettura/scrittura. La procedura di formattazione suddivide il dischetto in 80 tracce e 16 settori, ciascuno dei quali può contenere fino a 256 byte. La capacità complessiva di un dischetto a singola faccia è quindi di 320 Kbyte.

All'accensione del computer, dopo aver innestato il Plus 3, assieme al normale titolo d'apertura viene visualizzato il messaggio ACORN ADFS, sigla di "Advanced Disk Filing System". Il sistema di archiviazione su disco (DFS) dell'Electron somiglia notevolmente a quello del Micro BBC (molti comandi sono identici), ma presenta lievi differenze di funzionamento. Il Plus 3 è corredato da un disco *Welcome*, che contiene programmi dimostrativi simili a quelli della cassetta abbinata all'Electron e una serie di utility di grande importanza per il funzionamento del sistema di archiviazione su disco, tra le quali: BACKUP (che permette di copiare un intero dischetto), DIRCOPY (per il trasferimento di specifici file da un disco a un altro), BUILD (che consente di incorporare in un disco le routine BOOT di caricamento) e DUMP (che visualizza il contenuto di un file in caratteri ASCII e in notazione esadecimale).

Come il DFS del Micro BBC, le istruzioni dell'Electron per il dischetto sono precedute dal simbolo *, quindi alcuni comandi del SO (quali *LOAD, *RUN e *DELETE) sono già familiari a chi usa abitualmente il sistema di archiviazione su disco del Micro. Il comando di catalogazione del disco, CAT, visualizza in sequenza le stesse informazioni di quello del Micro BBC: il titolo del dischetto, il numero del drive, le opzioni disponibili, il nome della "directory" e quelli dei programmi ivi contenuti.

La principale differenza tra i due sistemi operativi risiede nel metodo di caricamento di nuovi dischetti. Quando l'utente inserisce il disco nell'unità del Plus 3, deve informarne il sistema operativo digitando il comando *MOUNT, che provoca il caricamento in memoria della CSD ("Currently Selected Directory"

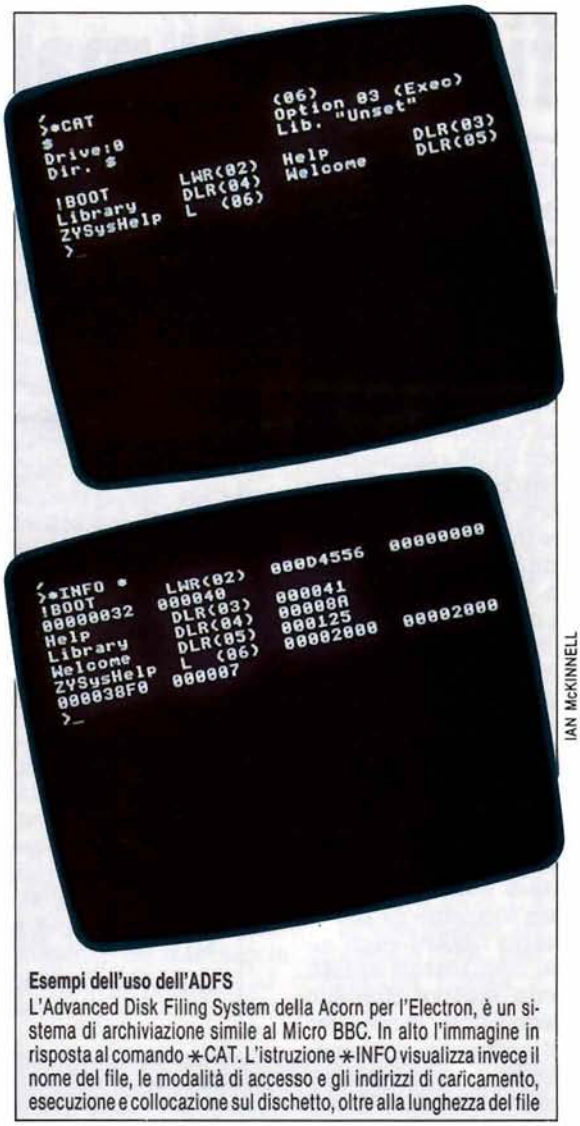


o "Directory Corrente") e della CSL (Currently Selected Library" o "Biblioteca Corrente"). Quando l'utente desidera cambiare dischetto, deve digitare il comando *DISMOUNT, che chiude tutti i file sequenziali eventualmente aperti e cancella dalla memoria la CSD e la CSL. Cambiato il dischetto, va impartito di nuovo il comando *MOUNT. A parte questi comandi, il DOS dell'Electron è identico a quello del Micro.

I file che compongono l'ADFS dell'Electron presentano un'organizzazione gerarchica che consente di accedere ai file in modo semplice e rapido, attraverso sequenze di nomi dette "pathname", ossia "nomi-percorso". Al vertice della struttura gerarchica si trova la "Directory Corrente" (CSD): quando l'apparecchio viene acceso, esso seleziona automaticamente la directory $, ossia la directory principale o "radice" . Per passare da questa ad una qualsiasi altra directory sul dischetto, l'utente deve digitare il comando *DIR, seguito dal relativo pathname. Alla CSD sono collegati una serie di file (o di directory secondarie), genericamente chiamati "file di biblioteca". Poiché la CSD può contenere un



CHRIS STEVENS



IAN McKINNELL

Esempi dell'uso dell'ADFS
L'Advanced Disk Filing System della Acorn per l'Electron, è un sistema di archiviazione simile al Micro BBC. In alto l'immagine in risposta al comando *CAT. L'istruzione *INFO visualizza invece il nome del file, le modalità di accesso e gli indirizzi di caricamento, esecuzione e collocazione sul dischetto, oltre alla lunghezza del file

ELECTRON PLUS 3

DIMENSIONI

mm. 365 x 200 x 50 (unità a dischetti)
mm. 365 x 90 x 50 (interfaccia)

CAPACITÀ

320 Kbyte su dischetti a singola faccia

INTERFACCE

Presa "jack" di alimentazione, bus di ingresso (del tipo per cartucce), bus di uscita su connettore a pettine

DOCUMENTAZIONE

Il manuale di corredo è molto chiaro e ben impostato; avvicina il principiante all'uso del dispositivo con un metodo graduale e progressivo. Alcune piccole imprecisioni potrebbero essere corrette

PRO

Ad un prezzo ragionevole, questo dispositivo amplia notevolmente le capacità dell'Electron

CONTRO

La quantità di software disponibile sul mercato è piuttosto scarsa e, per far uso di tutte le possibilità offerte da un sistema a dischetti, risultano necessari altri extra (quali un'interfaccia per stampante, contenuta nel Plus 1)

massimo di 47 file, questi devono essere organizzati in biblioteche: ciò consente di avere sul dischetto un gran numero di file. Le directory secondarie possono contenere i file che, per ragioni di spazio, non possono essere inserite nella directory principale CSD. Per accedere a un ipotetico file chiamato Archivio, caricato su un dischetto che contiene una serie di directory con biblioteche, occorre un comando del tipo: $.library1.archivio. Questa procedura può sembrare eccessivamente lunga, ma consente di avere sul dischetto più file, tutti chiamati Archivio, inseriti in diverse biblioteche. È possibile, inoltre, mantenendosi all'interno di una particolare biblioteca, correggere e salvare più volte il file senza influire sugli altri file omonimi del dischetto.

Anche l'ADFS dell'Electron, come il sistema d'archiviazione su disco del Micro BBC, offre numerose funzioni di utilità. Possiede una serie di comandi per cambiare il nome di dischi, di file e di biblioteche e altri per la gestione dell'accesso ai file. Il comando *ACCESS limita le possibilità di accesso ad un file: la sequenza *ACCESS file1 RL impedisce di "scrivere" su un file, evitando così involontarie cancellazioni, permettendone la sola lettura. Il comando *ACCESS file1 WR, invece, abilita sia la lettura sia la scrittura di un file. La lettera E, inserita dopo il nome del file, impedisce la copia di un file in codice macchina dal dischetto, limitando il numero e il tipo di comandi eseguibili sul file in questione.

Il manuale del modulo Plus 3 (esauriente e decisamente superiore alle documentazioni solitamente fornite dalla Acorn) contiene un completo "corso didattico", ma presenta alcune imperfezioni piuttosto fastidiose. Ad esempio, la descrizione delle sigle CSD e CSL, che compaiono per la prima volta a pag. 18, viene data solo alcune pagine dopo.

Fin dalla sua comparsa sul mercato, l'Electron è stato considerato dai più come il "parente povero" del Modello B del Micro BBC. La mancanza di interfacce, soprattutto di una porta per l'unità a dischetti, ha indotto molti a ritenerlo più un computer giocattolo che un vero apparecchio da lavoro. Tuttavia, grazie al notevole calo del suo prezzo e alla disponibilità delle interfacce Plus 1 e Plus 3, l'Electron sta lentamente riguadagnando la stima del pubblico.

Una sofisticata "mano"

Un nuovo progetto: un dispositivo per afferrare e spostare oggetti

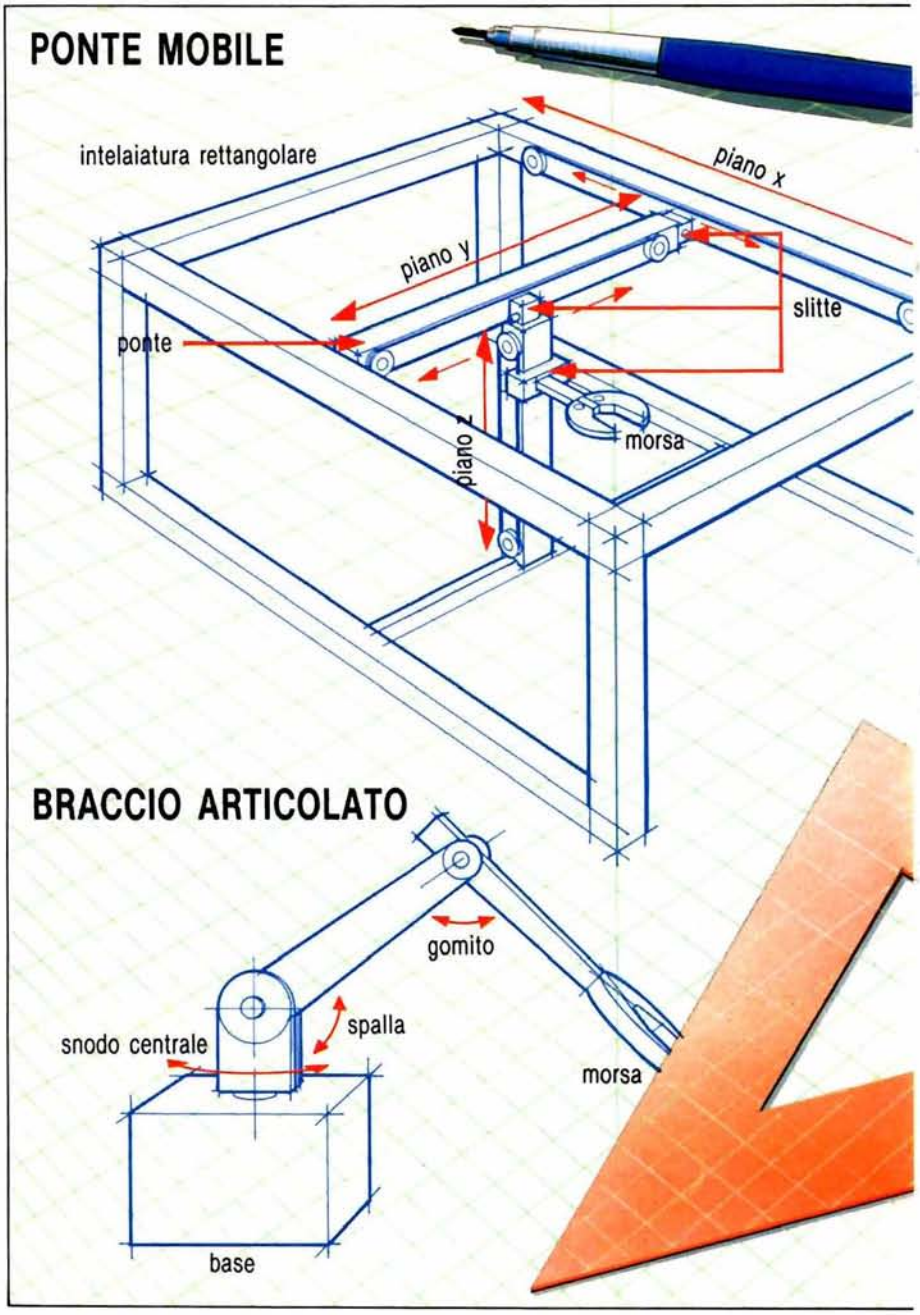
Di solito l'analisi di un progetto inizia dalla definizione del risultato che si vuol raggiungere. L'obiettivo del nostro lavoro è realizzare un apparecchio capace di muoversi da un dato punto, raccogliere un oggetto (all'incirca del peso e delle dimensioni di un rocchetto di filo) e tornare al punto di partenza con un margine d'errore inferiore ai 2,5 millimetri. Supponiamo di conoscere in anticipo le diverse posizioni della macchina. Un apparecchio industriale paragonabile al nostro progetto potrebbe essere un robot che inscatola oggetti in arrivo su un nastro trasportatore.

Una macchina con questi requisiti deve possedere numerosi motori idraulici o pneumatici ma, nel nostro caso, è previsto un sistema basato su motori elettrici.

Dei tre principali tipi di motori elettrici in commercio, il più semplice è il tipo a corrente continua. Questo motore è poco costoso, ma è difficile controllarlo con precisione tramite un'apparecchiatura digitale come un home computer. Per ottenere un margine di precisione adeguato, è necessario fissare sugli alberini di questi motori dei codificatori assiali (vedere pag. 1122). Ciascun motore richiede inoltre una logica di decodifica, tanto per l'hardware quanto per il software, che consenta un accurato controllo del posizionamento. I motori "passo passo" (vedere pagg. 1315-1317) e i servomotori a corrente continua risultano molto più facilmente controllabili da un computer, ma i primi sono piuttosto costosi, necessitano dei più disparati circuiti di comando e di solito sono troppo grandi per un progetto di questo tipo. Al contrario, i servomotori offrono due grossi vantaggi per un sistema in scala ridotta come il nostro: sono adatti allo scopo e, poiché vengono prodotti in gran quantità per il mercato dei modelli radiocomandati, sono facilmente reperibili e relativamente economici.

Il funzionamento dei servomotori è gia stato descritto in precedenza (vedere pagg. 1392-1394 e 1403-1405). Per ricapitolare si può dire che un servomotore è composto da un piccolo motore a corrente continua (5 volt), controllato da un circuito digitale a retroazione (feedback). La rotazione angolare dell'alberino del motore viene controllata da un piccolo potenziometro e confrontata all'interno del chip con il relativo segnale di comando. Quest'ultimo è un impulso la cui durata, compresa tra uno e due millisecondi, determina la rotazione angolare dell'alberino del motore. Gli impulsi di comando devono essere ripetuti con una frequenza di 50 Hz (cicli per secondo).

Il software di comando per i servomotori singoli e multipli è stato già descritto nella sezione LABORATORIO. Il programma prevede l'uso delle interrupt, che permettono al software in BASIC di funzionare normalmente mentre, a intervalli regolari, vengono inviate serie di impulsi ai motori.



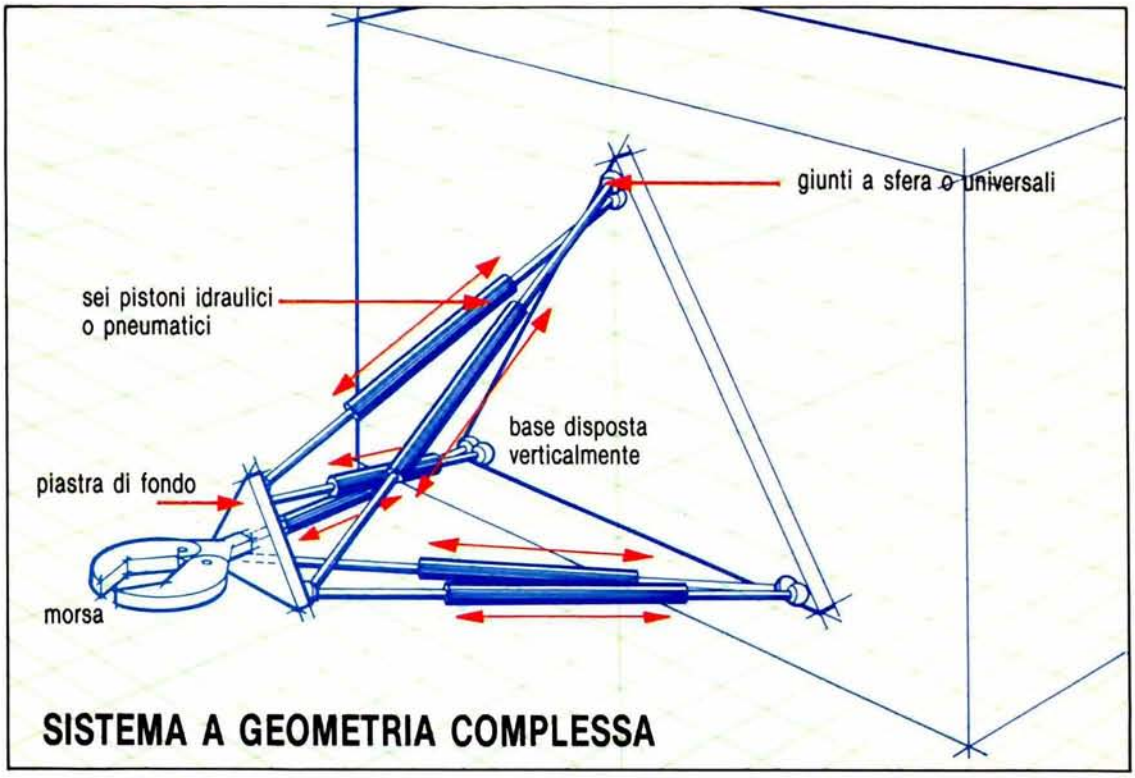
Possibili prototipi

La figura mostra due modelli alternativi, capaci di afferrare un rocchetto di filo e di inserirlo in un contenitore. Il ponte mobile, che possiede una struttura più semplice, non può eseguire lavori complessi come quelli di cui è capace il braccio robot. Quest'ultimo richiede un controllo molto più sofisticato anche per semplici spostamenti lineari del dispositivo di presa. Il controllo mediante computer consente di eseguire i calcoli necessari

Assetto dell'apparecchio

L'apparecchio può essere progettato in molti modi, ma occorre tener conto di alcuni requisiti fondamentali. Innanzi tutto serve un meccanismo di presa, azionato da almeno un motore. Secondariamente, il dispositivo di presa





KEVIN JONES

Geometria complessa

Il controllo computerizzato di motori elettrici, idraulici e pneumatici fornisce agli apparecchi robot una notevole elasticità d'impiego. Le complesse interazioni dei motori, che muovono i diversi segmenti della macchina, vengono controllate da un software piuttosto sofisticato di cui, spesso, l'utente ignora la complessità. Immaginiamo, ad esempio, le complicate manipolazioni consentite da questo sistema a sei pistoni, tutti controllati dal software di un computer

deve potersi muovere su tre piani. In media, l'apparecchio deve potersi spostare lateralmente di 10 cm. (piano x), frontalmente di pochi centimetri (piano y) e verticalmente di 10 cm. (piano z). Ciascun piano di movimento implica la presenza di un motore. Il dispositivo di presa viene manovrato azionando in modo combinato i diversi motori e deve possedere un margine di precisione di almeno 2,5 mm. per riuscire a tornare alla posizione di partenza. Inoltre deve essere in grado di sollevare un peso di almeno 20 grammi e le ganasce devono potersi aprire di 3 cm. per afferrare un oggetto delle dimensioni di un rocchetto di filo.

Un modello che risponde a questi requisiti potrebbe essere un piccolo ponte a slitte, sostenuto da una intelaiatura rettangolare: il ponte e la morsa poggiano su slitte collegate a pulegge che vengono azionate da motori passo passo o servomotori. Il sistema utilizza 4 motori: tre per controllare il movimento delle slitte; uno per il dispositivo di presa.

Un modello alternativo, il cui progetto si ispira al braccio umano, è un braccio articolato. La rotazione di ciascuna giuntura viene controllata da un solo motore. I servomotori digitali sono ideali per questa applicazione perché la loro rotazione può essere facilmente pilotata dal software del computer, consentendo un accurato controllo dei diversi segmenti del braccio. Rispetto al modello precedente, il posizionamento del dispositivo di presa richiede un programma di controllo più sofisticato. Per muovere, ad esempio, la morsa in un solo piano (supponiamo il piano x), occorre controllare simultaneamente almeno due motori del braccio. Nel modello a ponte, invece, è sufficiente azionare un solo motore. Tuttavia, poiché il sistema viene controllato da computer, è abbastanza facile ottenere complicate sequenze di controllo sviluppando un adeguato software di comando.

Il principale vantaggio offerto dal braccio articolato, rispetto al ponte, è la maggior flessibilità del sistema. Mentre quest'ultimo esegue solo il lavoro per cui è stato progettato e necessita di modifiche strutturali per poter svolgere altre operazioni, il braccio articolato può modificare il proprio modo di operare. Ad esempio, può inserire un rocchetto di filo in un contenitore posto al suo fianco. Conviene dunque orientare la scelta verso un braccio robot, del tipo mostrato nella figura, come modello base per il nostro progetto.

Indicazioni... a colori

In ciascuna fase del progetto, la sezione del braccio robot trattata nel testo viene evidenziata in colore



IAN McKINNELL

1 + 2 = 20 ?

La disciplina nella classificazione dei dati e le capacità aritimetiche del PASCAL

Quanto fa uno più due? In mancanza di ulteriori informazioni l'ovvia risposta è tre, dando per scontato che si stia operando con numeri naturali puri. Ma supponiamo di prendere una moneta da 10 lire e due da 5 lire: in questo caso il risultato non è tre, bensì 20 lire. Classificare i dati coi quali si opera è fondamentale per l'impostazione di qualsiasi problema e il linguaggio PASCAL aiuta ad organizzare e ad esporre in modo logico e comprensibile i dati da elaborare.
Come si è già visto, in PASCAL è possibile creare nuove ed originali categorie di dati, cosicché ci si può dedicare alla soluzione di un problema e allo sviluppo di algoritmi adatti al particolare tipo di dato trattato. Se, in un momento di distrazione, il programmatore inserisce nel listato un'operazione priva di senso (come cercare di leggere da tastiera un valore booleano), il compilatore del PASCAL è in grado di segnalarla, in quanto errore logico, facendo risparmiare ore di lavoro. L'incompatibilità tra i diversi tipi di dati impiegati e l'abbondanza delle procedure previste per descriverli è una delle peculiarità più significative del PASCAL. Questa caratteristica, ben lontana dal limitare le opportunità di programmazione, induce il programmatore a trarre il massimo vantaggio dalle diverse tipologie di dati offerte dal linguaggio e lo abitua a imporre ulteriori restrizioni alle proprie variabili.
Dopo aver illustrato l'uso di tutti i tipi di variabili semplici (ossia quelli che possono avere solo un valore), ricapitoliamo brevemente le regole cui essi sono soggetti. Nessuno scalare è compatibile con una variabile di diverso tipo scalare, anche se i due tipi numerici possiedono molte caratteristiche comuni e occasionalmente possono essere correlati. Poiché un numero reale può avvicinarsi per approssimazione ad un numero intero, è consentito assegnare numeri interi alle variabili reali, ma non viceversa. Vediamo alcuni esempi:

```
Program Compatibili (input, output);

VAR
    intA,
    intB :integer;
    Xreale,
    Yreale :real;

BEGIN
    read (intA,Xreale); {legge un intero, quindi un qualsiasi
        numero consentito}
    Yreale := intA; {reale := intero assegnazione consentita}
    intB := Xreale; {**ERRORE: assegnazione non consentita**}
```

È bene ricordare, anche se può sembrare superfluo, che le operazioni aritmetiche sono consentite soltanto su dati di tipo numerico. Per i numeri interi e reali vengono usati i soliti quattro operatori simbolici:

```
+ addizione
- sottrazione
* moltiplicazione
/ divisione
```

che, in questo contesto, si comportano come operatori binari o "diadici", poiché implicano sempre due operandi (interi o reali). Quando uno degli operandi è reale, il risultato dell'espressione è reale: il risultato di 2+2.0, ad esempio, è 4.0 e non 4. Nel caso della divisione, il quoziente è reale anche quando entrambi gli operandi sono interi: il risultato di 3/5 è 0.6, quello di 8/4 è 2.0.
Quando si dividono due interi, un risultato di tipo reale è spesso privo di senso. Se, ad esempio, 12 matite vengono suddivise tra 10 persone, si ottiene una matita a testa, con due di resto. Il risultato intero e il resto possono essere calcolati con i due operatori per la divisione tra interi DIV e MOD (entrambe parole riservate, in PASCAL): 15 DIV 5 = 3 e 15 MOD 5 = 0; 31 DIV 7 = 4 e 31 MOD 7 = 3. Attenzione ai valori negativi, però, in quanto la divisione tra interi non accetta numeri negativi al denominatore. In entrambi i tipi di divisione (reale e intera), il tentativo di dividere per zero produce sempre un errore in fase di esecuzione (run – time error), almeno finché qualcuno non inventerà un sistema per calcolare un risultato infinito!
All'interno di un'espressione, la moltiplicazione e la divisione vengono eseguite prima dell'addizione e della sottrazione. Per modificare quest'ordine di priorità si possono adoperare le parentesi tonde: ad esempio (8+4) DIV 2=6 mentre 8+4 DIV 2=10.

Funzioni di trasferimento

Sebbene il Pascal non consenta di assegnare direttamente un valore reale a un intero, è possibile ottenere lo stesso risultato ricorrendo alle funzioni di trasferimento trunc e round. La funzione trunc tronca un numero reale trascurando, se presente, la parte frazionaria: quindi, trunc (3.999)=3 e trunc (– 123,456)= – 123. Round esegue invece un arrotondamento "intelligente", portando la parte fratta a zero quando è minore di 0.5 e all'unità superiore quando è maggiore o uguale a 0.5. Ad esem-

pio, round (1234.5) = 1235 e round (— 0.49237) = 0. Poiché il PASCAL visualizza un messaggio d'errore, se il risultato di queste due funzioni su reali è un intero esterno alla gamma prevista dalla tipologia del dato (da — MaxInt a MaxInt), conviene verificare in precedenza la gamma di valori con un'istruzione IF. Si ricordi che l'oggetto delle due funzioni deve essere un numero reale e non intero (che, in quanto tale, è già troncato e arrotondato a sufficienza!).
Il PASCAL possiede una funzione chiamata odd insolita, per chi è abituato al BASIC. Questa funzione predefinita fornisce una risposta di tipo Booleano: "falso" se il dato in oggetto è un numero pari, "vero" se è un numero dispari. Ecco due esempi:

```
IF odd (N)
   THEN
      WriteLn ('È dispari!')
   ELSE
      WriteLn ('È pari!')
```

oppure:

```
IF odd (N) THEN
   IF N MOD 2=0 THEN
      WriteLn ('Getta via il compilatore!')
```

La tabella nella pagina seguente presenta tutte le funzioni aritmetiche del PASCAL che Queste possono essere applicate a qualsiasi dato numerico reale e intero. Le prime due, abs e sqr, restituiscono un valore compatibile con la tipologia del dato in esame: quindi, il risultato di abs(— 19.372) è 19.372, quello di abs(255) è 255. Le altre funzioni producono sempre un risultato reale, per cui il risultato di sqrt(16) è 4.0 e non 4, in quanto gli algoritmi di calcolo impiegati sono fondati sulla somma di una serie di valori, tutti frazionari.
Nel confronto di valori reali, non è mai del tutto preciso adoperare il simbolo di uguale, poiché anche il più piccolo errore di calcolo può portare due valori reali, nominalmente "identici", a differire. Una quantità di scarto può essere, ad esempio, sull'ordine di 1.0E—27: ai fini pratici è spesso insignificante, ma il computer non la riconosce come tale. Quindi, invece dell'istruzione IF X=Y THEN..., conviene usare la funzione predefinita abs del Pascal:

```
IF abs (X — Y) < (Il più piccolo valore significativo) THEN [ecc.]
```

Questa linea verifica se la differenza tra X e Y è minore del più piccolo valore significativo in un dato programma. Tale valore può essere specificato in una costante CONST all'inizio del programma.
In PASCAL, i logaritmi hanno la base naturale (e), non in base 10, mentre la funzione esp eleva e alla potenza indicata dal suo argomento. Per una deliberata scelta di Wirth, creatore del PASCAL, il linguaggio manca di un operatore per l'elevamento a potenza. Quante volte ci siamo imbattuti in programmi BASIC contenenti linee di questo tipo:

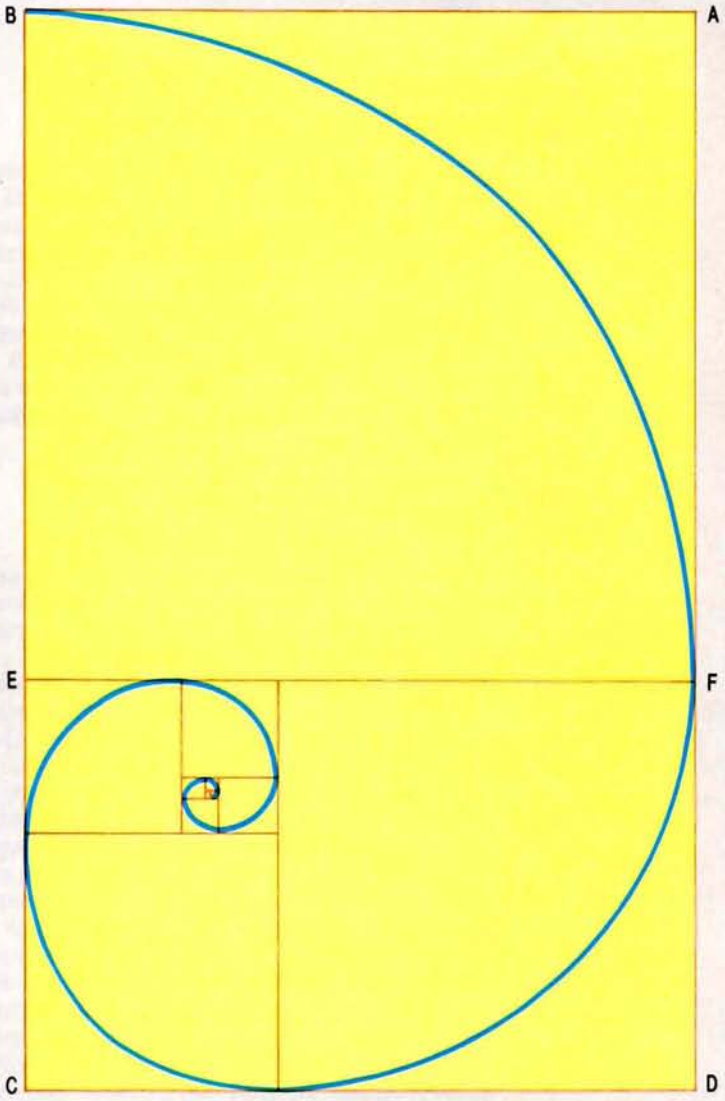
```
500 LET D=B + 4 * A * C
```

LA "SEZIONE AUREA"

Su una linea esiste un solo punto capace di suddividerla in due segmenti tali che il rapporto tra il segmento minore e quello maggiore sia identico al rapporto tra il segmento maggiore e l'intera linea. Tale rapporto viene detto "sezione aurea" e, come "pi greco", è un numero irrazionale, la cui parte decimale è composta da un numero infinito di cifre.



La sezione aurea, usata perfino da artisti e architetti dell'antica Grecia, viene oggi impiegata nella teoria delle proporzioni. Inoltre, alcune sue applicazioni compaiono anche in natura: la spirale delle chiocciole marine si sviluppa secondo la sezione aurea, dato che i suoi raggi vettori, separati da un angolo di 90°, presentano questo rapporto.



LIZ DIXON

I lati del rettangolo ABCD seguono il rapporto della sezione aurea. Al suo interno è possibile definire un secondo rettangolo ABEF, ottenendone un terzo (EFDC) i cui lati hanno lo stesso rapporto di quelli di ABCD. Il processo di suddivisione può continuare all'infinito; gli angoli dei quadrati sono tutti punti della "spirale della sezione aurea"

che usano una procedura lunga ed imprecisa, invece della semplice moltiplicazione B — B, per calcolare il quadrato di un numero? La funzione sqr (N) del PASCAL calcola il quadrato di N e dà come risultato un numero intero quando N è un intero e un numero reale quando N è reale (ad esempio, sqr (X/3)). Infine la funzione sqrt() calcola la radice quadrata di un numero.

Sottotipi

Tutti i tipi di scalari possiedono una gamma definita e ordinata di valori e possono essere a loro volta suddivisi in nuovi sottotipi. Quando viene definito l'identificatore di un nuovo sottotipo, i suoi limiti inferiore e superiore vengono indicati intercalandoli con due punti (..). Ad esempio:

```
TYPE
    byte = 0 .. 255; {sottotipo di integer}
    alfa = 'A' .. 'z'; {sottotipo di char}
    semi = (picche, cuori, quadri, fiori); {nuovo tipo}
    maggiore = picche .. cuori; {sottotipo di semi}
```

A parte i vantaggi descritti in precedenza, un buon compilatore PASCAL, capace di ottimizzare il codice, immagazzina le variabili di tipo byte in 8 bit, anziché nei 32 previsti (se questo è il formato normale). Quindi una matrice di grosse dimensioni occupa solo il 25% della memoria impiegata da elementi interi non ottimizzati. Si noti che il tipo regina ha due possibili valori, piuttosto simili a quelli Booleani che vengono definiti nel seguente modo:

```
TYPE
    boolean = (false,true);
```

Tutti i sottotipi ereditano la classificazione dei dati definita per i loro tipi d'origine. La precedente definizione di alfa non riguarda esclusivamente le lettere alfabetiche. In particolare, il set di caratteri ASCII prevede, tra "Z" e "a", sei simboli di diverso tipo, compresi i due simboli di parentesi quadra ([e]). Questi fanno parte dei 23 simboli "riservati" del PASCAL e, come avviene in altri linguaggi, vengono usati per delimitare i componenti di alcuni tipi di dati strutturati. Inizialmente il BASIC prevedeva l'uso di parentesi quadre per racchiudere gli indici di una matrice ma, in seguito, venne deciso di sostituirle con parentesi tonde, poiché i primi computer talvolta adottavano sottogruppi di caratteri ASCII, privi di lettere minuscole, parentesi quadre, parentesi graffe (usate per delimitare i commenti PASCAL) e di altri simboli. Lo stesso Apple II, non modificato, risente di queste carenze. Nei casi in cui si presenta questo problema, il PASCAL offre come alternativa alle parentesi quadre e graffe i rispettivi formati (. e .) e (— e —).
La seconda coppia di simboli viene comunemete usata per delimitare i commenti. Tuttavia, solo i compilatori approvati dalla ISO dispongono, in genere, di simboli alternativi per le parentesi quadre. In questi compilatori è quasi sempre possibile effettuare una sola o entrambe le sostituzioni, quindi: (—questo è un commento consentito).

Funzione	Valore calcolato
abs (K)	Valore assoluto di K
sqr (K)	Quadrato di K
sqrt (K)	Radice quadrata di K
sin (A)	Seno di A (in radianti)
cos (A)	Coseno di A (in radianti)
arctan (T)	Angolo (in radianti) della tangente T
ln (K)	Logaritmo naturale di K
exp (L)	e elevato a potenza L (anti—ln di L)

LA "SERIE" DI FIBONACCI

Il seguente programma calcola i numeri di Fibonacci e li visualizza assieme al rapporto che li lega. Il programma evidenzia una delle proprietà della serie di Fibonacci (1,1,2,3,5,8,13, ecc.), ossia: il rapporto che intercorre tra ciascuna coppia di numeri successivi cresce approssimativamente del valore della sezione aurea.
Il programma dimostra alcuni dei principi descritti in precedenza, compresi gli esempi della struttura REPEAT del PASCAL. Come esercizio, si provi a modificare la condizione di uscita dal ciclo, in modo che il programma termini quando il successivo numero di Fibonacci da calcolare è maggiore di MaxInt

```
PROGRAM  Aurea          (output);

CONST
         Epsilon                    = 1.0E—06;

TYPE
         Fibonacci                  = 1 .. MaxInt;

VAR
         primo,
         secondo,
         succ           : Fibonacci;
         rapp,
         Oro            : real;
         conto          : integer;
BEGIN
    WriteLn ('La Sezione Aurea' : 30);
    WriteLn;
    WriteLn ('Serie di Fibonacci :');
    primo := 1; {per definizione}
    secondo := 1;
    rapp := primo/secondo;
    conto := 0;

    REPEAT
        IF conto MOD 10 = 0 THEN
            BEGIN {testazione ogni 10 linee}
                WriteLn;
                WriteLn ('Primo' : 10,
                         'Secondo' : 10, 'Rapporto' : 14);
                WriteLn
            END;

        WriteLn (primo : 10, secondo : 10, rapp : 16 : 8);
        conto := conto + 1;
        Oro := rapp; {ricordare la vecchia S.A.}
        succ := primo + secondo;
        primo := secondo; {incrementa ...}
        secondo := succ; {La serie}
        rapp := primo/secondo

    UNTIL abs (rapp - Oro) < Epsilon;

    WriteLn;
    WriteLn ('La Media Aurea è : ', 100 — rapp : 10 : 5, '%')
END .
```

Interrompiamo il programma...

La gestione delle interrupt consente al SO di distribuire le proprie capacità elaborative

Quando si esegue un programma BASIC sul Micro BBC, si può avere l'impressione che l'apparecchio venga coinvolto per intero nell'elaborazione del programma. In realtà, il suo sistema operativo svolge contemporaneamente molte altre attività: tra queste, ad esempio, la scansione della tastiera. L'impiego (solo apparentemente simultaneo) delle risorse del computer è reso possibile dalle interrupt. In breve, una interrupt è un segnale che ordina alla CPU di interrompere quanto sta facendo e di eseguire un altro compito. Al termine, la CPU riprende il lavoro interrotto come se nulla fosse accaduto. Sul Micro BBC, le interrupt controllano svariate attività (la lettura della tastiera, il lampeggiamento dei colori, l'elaborazione del comando ENVELOPE e la lettura del convertitore A/D). Esaminiamo più dettagliatamente come l'apparecchio esegue queste operazioni.

Le interrupt del Micro BBC vengono generate da varie componenti hardware, tra le quali il chip dell'interfaccia seriale, i VIA (Versatile Interface Adaptors) di sistema e dell'utente, le interfacce per i dischetti e l'Econet. Questi dispositivi trasmettono due diversi tipi di interrupt al processore 6502 che interrompono entrambi il lavoro del processore ma, in caso di necessità, è possibile ordinare alla CPU di ignorare le interrupt chiamate IRQ ("Richieste d'Interruzione Mascherabili"). Il secondo tipo di interrupt, che la CPU non può in alcun caso ignorare, è quello delle interrupt ad alta priorità, chiamate "Interrupt Non Mascherabili" o NMI.

Nel Micro BBC le NMI vengono prodotte da dispositivi dell'hardware che richiedono un'attenzione immediata da parte della CPU, mentre le IRQ da componenti a priorità inferiore. In effetti, gli unici due dispositivi del BBC che usano le NMI sono la rete Econet e il sistema d'archiviazione su disco. Quando riceve una interrupt di questo tipo, il SO esegue una breve routine in codice macchina memorizzata alla pagina &0D. Quest'area è stata scelta perché risulta inaccessibile all'utente, quando al computer è collegata un'unità a dischetti.

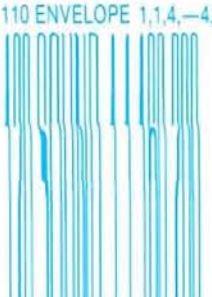
Dato che il programmatore viene diffidato dall'usare le NMI (per non compromettere il funzionamento del sistema), soffermarsi su di esse ha scarsa utilità pratico. Al contrario, conviene esaminare a fondo il funzionamento delle interrupt IRQ sul Micro BBC. Le interrupt mascherabili per il processore 6502 possono essere disabilitate tramite l'istruzione assembler SEI e riabilitate con CLI. Le IRQ sono diverse a seconda del dispositivo che ha generato l'interrupt. Per prima cosa la CPU si occupa di individuarne la provenienza interrogando in successione i vari componenti dell'hardware capaci di produrre IRQ. Poi, dopo aver eseguito la routine abbinata all'interrupt, il dispositivo che l'ha generata riceve l'ordine di dimenticare l'interrupt, finché non si verifichi un altro evento che giustifica l'invio di un'ulteriore IRQ al 6502. Questa procedura viene detta "cancellazione" dell'interrupt, mentre le routine in codice macchina che gestiscono queste interrupt sono chiamate ISR (Interrupt Service Routine).

Funzionamento di una ISR

Quando il programma viene interrotto, i valori contenuti nei registri X e Y del 6502 non devono subire alcuna alterazione, quindi il primo compito di una routine ISR è quello di salvare sullo stack del 6502 il contenuto dei registri, che va recuperato al termine della routine medesima. La routine passa quindi all'elaborazione della IRQ e rimuove la condizione d'interruzione, prima di restituire il controllo al programma interrotto. Il motivo di questa cancellazione è abbastanza ovvio: se, quando il controllo torna al programma interrotto, l'interrupt non è stata rimossa, il 6502 la "servirebbe" nuovamente, entrando così in un ciclo senza fine. Dopo aver genericamente descritto la gestione di un'interrupt, esaminiamo il caso particolare del Micro BBC. Questo computer usa le interrupt per un gran numero di operazioni: va fatta attenzione a non disabilitarle, con l'istruzione SEI, per un tempo superiore a pochi millisecondi, altrimenti si potrebbe intralciare il corretto funzionamento del SO. Il seguente programma mostra cosa accade al Micro BBC quando le interrupt vengono disabilitate per un dato tempo:

```
 10 MODE 2
 20 DIM C (100): REM riserva un'area per il codice macchina
 30 FOR I% = 0 TO 2 STEP 2
 40 P% = C
 50 [OPT I%
 60 .codice SEI /disabilita le interrupt
 70 RTS
 80 ]:NEXT
 90 COLOUR 14
100 PRINT "SALVE!"
110 ENVELOPE 1,1,4,-4,3,10,20,20,127,0,0,-5,126,126
```





```
140 REPEAT
150 PRINT TIME,I%
160 I% = I% + 1
170 IF TIME>100 THEN CALL codice
180 UNTIL FALSE: REM prosegue all'infinito
```

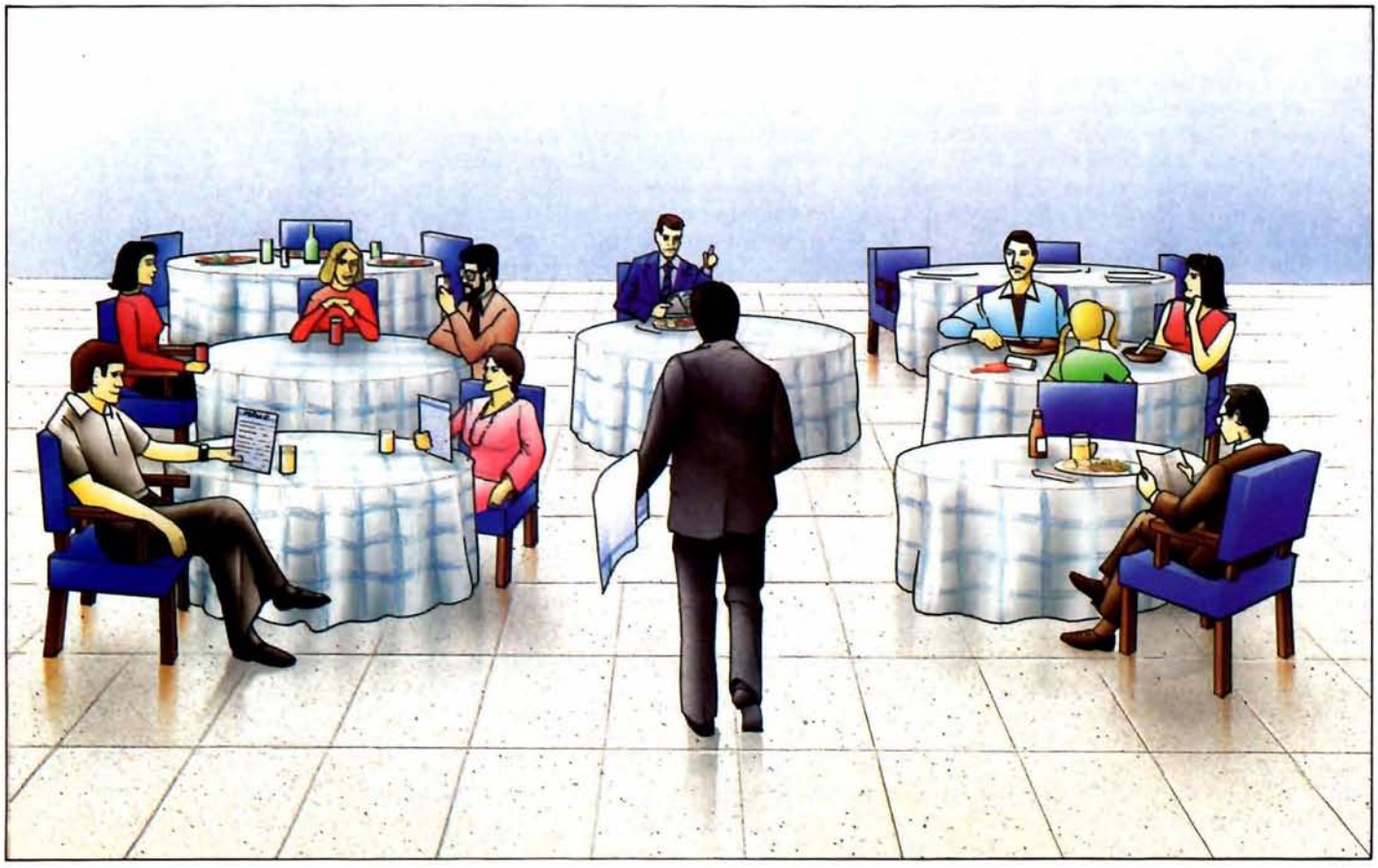
Eseguendo il programma ci si accorge che, quando le interrupt vengono disabilitate dalla chiamata alla routine codice in assembly, la variabile TIME non viene più aggiornata e si interrompe sia la visualizzazione dei colori lampeggianti sia l'elaborazione dell'istruzione ENVELOPE. Tuttavia l'apparecchio non si blocca, poiché la routine continua a visualizzare correttamente I%. Per porre fine al programma è necessario premere CTRL BREAK.

Cameriere!!

La CPU del 6502 può essere paragonata a un cameriere che riceve le ordinazioni in un ristorante. Così come il cameriere deve seguire con maggior attenzione i clienti, che hanno la priorità, e contemporaneamente sparecchiare i tavolini, preparare conti e via dicendo, anche la CPU deve prestare costante attenzione alle periferiche collegate, controllando lo scambio d'informazioni e l'esecuzione del programma.
Un buon cameriere deve seguire tutti i clienti e portare a termine tutti i lavori lasciati momentaneamente in sospeso, assicurandosi che le ordinazioni arrivino rapidamente ai vari tavoli. La CPU si comporta in modo simile, rispondendo alle interrupt o agli eventi generati dall'hardware o dal software di controllo. Dopo aver elaborato un'interrupt, la CPU può riprendere il lavoro interrotto

Il vettore IRQV1

Quando il 6502 riceve una IRQ, il controllo passa alla routine ISR il cui indirizzo è contenuto in un vettore chiamato IRQV1. L'indirizzo di questo vettore è &204 e &205. La ISR puntata dal vettore provvede ad individuare il dispositivo, tra quelli previsti dal SO, che ha inviato l'interrupt. La routine ISR indirizzata dal vettore IRQV1 gestisce le IRQ generate da tre componenti hardware: il chip dell'interfaccia seriale, il VIA di sistema e parte del VIA dell'utente, in quest'ordine di priorità.
Il chip dell'interfaccia seriale, ossia il dispositivo di maggior priorità che usa il vettore, controlla l'interfaccia RS423 e quella per il registratore a nastro: le interrupt che esso genera riguardano ovviamente queste due interfacce.
Il chip del VIA di sistema, che collega la CPU a una serie di altri dispositivi del computer e fornisce alcune funzioni di temporizzazione, genera interrupt di priorità immediatamente inferiore. Le IRQ prodotte quando viene premuto un tasto, ad esempio, comunicano al SO che deve elaborare un'informazione proveniente dalla tastiera. Altre interrupt vengono generate durante l'invio di immagini al monitor o al televisore e vengono prodotte a una frequenza di 1/50 di secondo. Queste sono dette "Interrupt di Sincronizzazione Verticale".
Anche il convertitore A/D interagisce con il chip del VIA di sistema, che genera un'interrupt al termine di ogni conversione di un segnale. Questa interrupt induce il SO ad aggiornare i valori usati per comunicare valori numerici alla funzione ADVAL. Un'altra importante interrupt, prodotta dal VIA di sistema a una frequenza di 1/100 di secondo, porta il SO a incrementare la variabile TIME, ad aggiornare il "temporizzatore d'intervallo" (usato nella generazione degli "eventi"), a decrementare il temporizzatore INKEY (che determina i tempi di ritardo per un'istruzione INKEY) e, infine, a elaborare parte di una istruzione ENVELOPE. Inoltre induce il SO a proseguire l'elaborazione dei tasti che vengono premuti. Altre interrupt generate dal VIA di sistema riguardano il sistema di sintetizza-



KEVIN JONES

zione vocale e l'impiego della penna ottica. Sebbene esistano alcune chiamate OSBYTE che consentono di modificare la procedura seguita dal SO nel rispondere alle interrupt generate dai chip dell'interfaccia seriale e dal VIA di sistema, è bene ricordare che conviene manipolare il vettore IRQV1 solo sapendo con estrema precisione cosa si sta facendo.
Le interrupt di minore priorità, trattate da questo vettore, originano dal VIA dell'utente, che controlla il funzionamento delle porte utente e per la stampante.
Le interrupt non gestite dal vettore IRQV1 vengono passate a una ISR puntata dal vettore chiamato IRQV2: la maggior parte di queste provengono dal VIA dell'utente, purché programmato per produrre IRQ destinate alla CPU, altrimenti non vengono normalmente riconosciute dalla CPU.

Il vettore IRQV2

Il vettore IRQV2, contenuto nelle locazioni &206 e &207, può essere modificato per puntare a una routine ISR scritta dallo stesso utente. Sebbene sia possibile modificare anche IRQV1, per chiamare le nuove routine, è sempre preferibile ricorrere al vettore IRQV2. La procedura di modifica di un vettore, affinché esso punti a una nuova routine, viene talvolta chiamata "intercettamento" del vettore. La routine chiamata tramite questa tecnica di intercettamento del vettore IRQV2 deve soddisfare le seguenti condizioni:

- La CPU deve avere già salvato nella locazione &FC il contenuto del registro X ed appoggiato sullo stack il registro di stato del processore (PSR).
- Il contenuto dei registri X e Y, prima della comparsa dell'interrupt, deve essere ripristinato al termine di essa.

Per rispettare la seconda condizione, i registri X e Y vanno salvati al momento della chiamata della routine ISR con i seguenti comandi:

```
TXA:PHA
TYA:PHA
```

Inoltre, la routine dell'utente deve eliminare la condizione d'interruzione. La procedura di cancellazione dipende dal componente che ha generato l'interrupt e, nel caso di una interrupt prodotta dal VIA dell'utente, implica la lettura o la scrittura di particolari registri del chip VIA. Prima di riprendere il programma interrotto, occorre ripristinare i registri e usare un'istruzione RTI (non la semplice RTS), per concludere la routine di servizio. Un'ultima raccomandazione: se al momento della chiamata della routine (tramite il vettore IRQV1) il registro A viene salvato alla locazione &FC, la comparsa di una ulteriore interrupt potrebbe provocare la sovrascrittura della locazione, prima che venga conclusa l'elaborazione della prima interrupt (quella dell'utente). Esistono due soluzioni: inserire un'istruzione SEI all'inizio della routine e una CLI prima dell'uscita, oppure appoggiare il registro A sullo stack al momento della chiamata e recuperarlo al termine della routine. In questo caso è necessario estrarre il valore del registro A da &FC, prima di eseguire l'istruzione RTI.
Il Micro BBC, come alternativa alle interrupt, usa "segnali di evento", che formano l'oggetto della prossima trattazione.

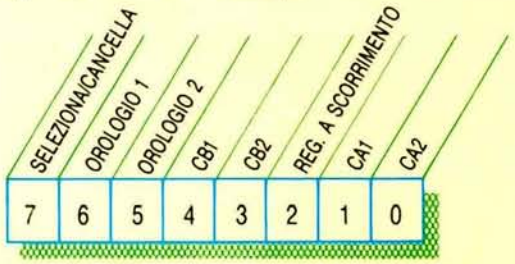
LE INTERRUPT DEL CHIP VIA

LIZ DIXON

Registro di segnalazione delle interrupt



Registro di abilitazione delle interrupt

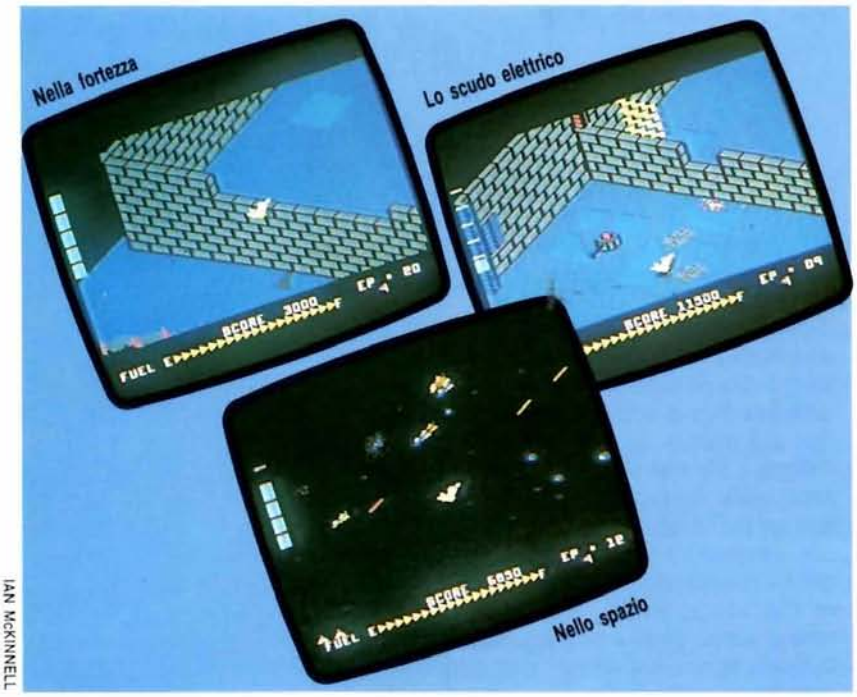


Benché il 6502 del Micro BBC possieda solo due linee d'interruzione, IRQ e NMI, è possibile programmare i chip del VIA d'interfacciamento in modo da consentire a più periferiche di usare in comune la linea d'interruzione IRQ. Ogni chip possiede due porte di I/O a otto bit, ciascuna collegata a otto linee per i dati e a una coppia di linee di segnalazione, chiamate CA1 e CA2 sulla porta A e CB1 e CB2 sulla porta B. Tali linee vengono spesso usate per sincronizzare lo scambio d'informazioni tra il computer e le periferiche, ma possono funzionare anche come linee d'interruzione. Oltre ad esse, ciascun chip presenta un altro tipo di interrupt interna, generato dal proprio registro a scorrimento. Questo registro viene usato per trasmissioni seriali sulla linea CB2, in risposta a segnali provenienti dall'esterno sulla linea CB1. Infine, ogni chip possiede due orologi (timer) a 16 bit, capaci di generare interrupt. Tutte queste interrupt condividono la stessa linea IRQ del microprocessore. Quando il processore riceve un'interrupt, ne individua la provenienza esaminando il registro di segnalazione delle interrupt (Interrupt Flag Register o IFR), interno al VIA. Ogni bit di questo registro corrisponde a una particolare sorgente d'interrupt e viene posto a uno quando l'unità associata genera un'interrupt. Unica eccezione è il bit 7, che viene selezionato qualunque sia l'unità chiamante. In effetti, il bit 7 corrisponde a una linea IRQ. Può essere necessario ordinare alla CPU di trascurare le interrupt emesse da una particolare unità mentre ne sta servendo un'altra; ciò è possibile programmando adeguatamente il registro di abilitazione delle interrupt (Interrupt Enable Register o IER). Questo possiede 8 bit che corrispondono alle stesse unità abbinate al registro precedente. In simili casi, il processore presta attenzione solo alle interrupt generate dalle unità i cui bit di segnalazione sono posti ad uno

Zaxxon

Inizialmente disponibile soltanto nelle sale giochi, è diventato un classico per micro

Nel gioco *Zaxxon* (che è stato presentato sommariamente a pag. 768), il giocatore deve pilotare un'astronave e superare una serie di ostacoli nel tentativo di distruggere il robot Zaxxon.
All'inizio l'astronave si trova nelle profondità dello spazio, ma improvvisamente le compare di fronte una fortezza spaziale, protetta da un alto muro.
Per procedere, il giocatore deve pilotare l'astronave attraverso un piccolo varco del muro, quindi affrontare una serie di ostacoli che gli sbarrano il cammino, tra i quali i missili, i riflettori parabolici dei radar e i depositi di carburante, tutti rappresentati con una realistica grafica tridimensionale. I depositi sono particolarmente importanti: se il giocatore riesce a distruggerli, aumenta le scorte di carburante della propria astronave.
La grafica del gioco è molto curata: l'astronave è rappresentata mediante tre diversi sprite (uno per il volo normale, uno per le cabrate e uno per le picchiate), le cui dimensioni crescono o decrescono a seconda della manovra effettuata.
Anche gli ostacoli sullo sfondo sono molto particolareggiati e la loro prospettiva accresce il realismo del gioco. L'astronave proietta sugli oggetti dello sfondo un'ombra che diviene più grande quando il velivolo perde quota.



IAN McKINNELL

L'ultima frontiera
Le immagini derivano da tre diverse fasi del gioco. Il giocatore inizia pilotando un'astronave all'interno di una prima fortezza: la nave deve cabrare e raggiungere una quota che le consenta di superare il muro. All'interno della fortezza c'è una seconda barriera, difesa da un campo elettrico. Se il giocatore colpisce le torrette dei cannoni, guadagna punti extra. Quando scompare la prima fortezza, la nave torna nello spazio, dove deve colpire (o perlomeno evitare) le navi nemiche che la attaccano. Infine, dopo aver superato le difese della seconda fortezza, l'astronave affronta il terribile robot Zaxxon

Se, in questa prima fase, la nave viene distrutta dal fuoco nemico, il giocatore deve riprendere il gioco dall'inizio. Tuttavia, con un po' d'esperienza, la maggior parte dei pericoli diventa facilmente superabile. Al giocatore conviene concentrare il proprio fuoco sui riflettori parabolici dei radar, poiché questi lanciano verso l'astronave veloci missili con traiettorie orizzontali. Inoltre occorre tenere sotto costante controllo la quota della nave: anche un piccolo errore di valutazione può portare alla collisione con un missile. Infine l'astronave deve schivare, cabrando, un campo di forze elettriche e tornare ad abbassarsi per poter annientare i "caccia" nemici, prima che essi decollino. Questa fase di gioco è molto delicata perché maggiore è il numero di aerei distrutti al suolo, minore sarà quello dei caccia che in seguito attaccheranno l'astronave.
Sbaragliate con successo le difese della fortezza, la nave torna nello spazio, dove viene attaccata da uno stormo di apparecchi avversari.
A questo punto il gioco tridimensionale prosegue su uno schermo a due dimensioni. Per poter colpire il nemico, il giocatore deve cabrare o picchiare con grande precisione; errori di valutazione sono abbastanza frequenti, poiché l'unico punto di riferimento sono le dimensioni relative delle diverse astronavi. Quella del giocatore raggiunge una corretta posizione di fuoco solo quando ha circa le stesse dimensioni del vascello nemico: a questo punto ha la meglio chi spara per primo!
Se l'astronave del giocatore esce vittoriosa dallo scontro, torna nuovamente alla fortezza spaziale. Stavolta però l'apertura nel muro è più piccola e gli obiettivi sono più difficili da colpire, per non parlare del robot Zaxxon, che avanza muovendosi lungo una griglia.
A questo punto la nave si ferma, mentre il robot Zaxxon avanza inesorabilmente: l'unica via di scampo è centrare con tre proiettili il lanciamissili dell'automa, pena l'annientamento completo!

Zaxxon: Per lo Spectrum Sinclair e il Commodore 64
Editori: US Gold, (GB)
Autore: Peter Adams
Joystick: Necessario
Formato: Cassetta (Spectrum, Commodore); dischetto (solo Commodore)



TONY SLEEP

Una bella idea

Gli allievi di questa scuola elementare inglese hanno appreso i principi elementari della geometria usando una tartaruga Valiant. Come illustrato nella foto, essi hanno attivamente partecipato alla costruzione di un labirinto e alla stesura del programma che permette alla tartaruga di evitare gli ostacoli. Questo originale esercizio ha facilitato la comprensione dei principi di rotazione angolare e di altri concetti astratti, che spesso i bambini trovano incomprensibili, o quanto meno privi d'interesse

L'allievo comanda

Oltre ad essere istruttivi, i robot da pavimento sono anche divertenti da usare

L'uso di robot nell'ambito didattico, specialmente per gli studenti più giovani, è un tentativo di affrontare in modo nuovo l'insegnamento di concetti non riferibili a modelli concreti. Usando una tartaruga, per esempio, è possibile dimostrare visibilmente i complicati principi della trigonometria oppure rappresentare i processi mentali di apprendimento. Inoltre i robot sono di grande aiuto nell'insegnamento delle basi della programmazione di un computer.

In passato, per imparare la geometria si usavano matita, righello e goniometro. Un ragazzo di 7 o 8 anni non vedeva alcun motivo pratico per imparare a misurare un angolo, né tantomeno riusciva a capire effettivamente i vari teoremi ad esso connessi. Al contrario, gli angoli assumono un significato molto concreto quando vengono usati per determinare se una tartaruga, programmata dallo studente, urterà o no le bottiglie di plastica allineate sul pavimento dell'aula scolastica. I principi della geometria forse continuano a rimanere avvolti da nebbia ma, almeno, c'è un motivo concreto per cercare di capirne le applicazioni pratiche. La differenza tra un angolo di 5 e uno di 8 gradi diviene evidente se, come risultato, la tartaruga urta una bottiglia o prosegue il suo cammino.

Uno dei principali aspetti dell'uso didattico della tartaruga è che permette di apprendere in modo empirico attraverso l'esperienza diretta. Un bambino può correggere più volte il programma finché non ottiene i movimenti desiderati e, contemporaneamente, acquisisce nozioni sulle proprietà degli angoli, delle linee e delle misure. Uno spostamento errato della tartaruga induce l'allievo a cercare l'errore per correggerlo e non a pensare che qualcosa ha funzionato male. Nella programmazione di un braccio robot, per sollevare e spostare oggetti, egli è addirittura costretto a ragionare in termini di movimenti tridimensionali. Questo risultato è difficilmente raggiungibile lavorando su un foglio di carta, limitato dalla sua bidimensionalità, mentre un braccio robot mette chiaramente in evidenza i tre piani di movimento. Grazie ai robot, i risultati della programmazione sono fisicamente visibili ed è molto più facile individuare e correggere gli errori di misurazione degli angoli e delle linee; inoltre lo studente può così valutare da solo il proprio lavoro.

Un giocattolo robot, comunemente usato ne-

ESEMPIO CONCRETO

Spesso i bambini hanno difficoltà a comprendere concetti astratti, anche se illustrati con esempi concreti. In quesa dimostrazione, che riguarda la conservazione delle quantità e le proprietà transitive, al bambino vengono mostrati 4 contenitori di forma diversa, ma di uguale capacità. A, B e C contengono la stessa quantità di liquido e i contenitori B e B' sono identici. Dopo aver travasato il liquido da un contenitore ad un altro, l'insegnante chiede al bambino quale pensa sia il risultato.



Lo sperimentatore versa il contenuto di A in B' e fa osservare al bambino che il livello del liquido in B e B' è identico. Quindi torna a travasare B' in A



La stessa operazione viene ripetuta per i contenitori C e B'



I contenitori A e C vengono affiancati e l'insegnante chiede al bambino se uno dei due recipienti contiene più liquido o se entrambi contengono la stessa quantità



I contenitori A, B e C vengono posti sul tavolo e al bambino viene chiesto se i tre recipienti contengono lo stesso volume di liquido. Se l'allievo risponde esattamente lo sperimentatore lo invita a spiegare il ragionamento seguito. Generalmente i bambini con meno di 7 anni non sono in grado di comprendere i principi alla base della dimostrazione o di dare una risposta corretta. Questo ed altri esperimenti dello stesso tipo, ideati da Piaget, hanno influenzato lo sviluppo del linguaggio LOGO. L'uso didattico dei robot fornisce ai bambini esempi concreti di concetti astratti, consentendo loro una più attiva partecipazione ai processi d'apprendimento



KEVIN JONES

gli asili e nelle scuole elementari inglesi, è il Big Trak, un futuristico veicolo dotato di una piccola memoria interna e di una tastierina tipo calcolatrice sulla parte superiore. Per determinare la direzione e la lunghezza del movimento basta premere uno dei 4 tasti a freccia e digitare un numero compreso tra 1 e 99. Il comando AVANTI 1 fa avanzare il veicolo lungo un percorso pari alla sua lunghezza; DESTRA 15 lo fa ruotare a destra di "15 minuti" (con riferimento all'orologio), ossia di 45°. Il robot, che accetta sequenze composte da un massimo di 8 comandi, viene usato al posto della tartaruga e incoraggia i bambini a programmare semplici routine. Come accade sui computer, il Big Trak esegue i comandi secondo l'ordine d'inserimento.

Un oggetto per pensare

Le tartarughe possono essere d'aiuto nell'insegnamento della matematica, ma la loro principale funzione è insegnare ai bambini come si programma un computer. Seymour Papert, l'inventore della tartaruga, (pag. 986), ha più volte affermato che deve essere il bambino a programmare il computer e non viceversa. Questa sua filosofia si riflette nel linguaggio da lui ideato, il LOGO, sviluppato appositamente per l'insegnamento. Secondo Papert, il bambino deve apprendere le basi della programmazione usando il LOGO per controllare i movimenti della tartaruga.
Paul Cheung, dell'Università di Edimburgo, ha sviluppato l'aspetto robotico del LOGO, dopo aver intuito che era necessario creare un software più versatile, in risposta ad un hardware sempre più sofisticato. Il LOGO, tuttavia, è un linguaggio ideale per il controllo di tartarughe, ma non di robot. Cheung desiderava offrire ai bambini la possibilità di controllare robot di diverso tipo; ha quindi ristrutturato il linguaggio, mettendolo in condizione di gestire un numero di funzioni maggiore accrescendone la capacità di ricevere informazioni da strumenti come sensori, lettori di codici a barre e rivelatori luminosi. Il risultato del suo lavoro è stato il Concurrent-LOGO, o C-LOGO, un linguaggio capace di eseguire "elaborazioni parallele", ossia di controllare contemporaneamente fino a un massimo di 8 operazioni.
Due funzioni particolarmente interessanti del C-LOGO sono FOREVER e WHENEVER. La prima avvia una procedura e la esegue finché il C-LOGO non la conclude. La seconda attiva una funzione solo quando si verifica una particolare condizione ed è alla base della elaborazione parallela. Inoltre il C-LOGO permette di accendere e spegnere interruttori e di verificare quali di questi sono accesi e quali no. Questa capacità viene sfruttata per controllare i sensori: quando un sensore di contatto urta un ostacolo, il relativo interruttore viene chiuso, comunicando così l'evento al computer. Il bambino può quindi programmare un robot affinché eviti ostacoli, servendosi della funzione WHENEVER per ordinare al

robot di cambiare direzione quando avviene un contatto.
Il C-LOGO è un linguaggio più adatto agli studenti delle scuole superiori, per i quali Cheung ha sviluppato software dedicato a svariati strumenti di controllo. Ragazzi di età compresa tra 13 e 15 anni hanno scritto numerosi programmi per controllare piccoli mulini a vento, elevatori, bracci meccanici e veicoli giocattolo o per interfacciare una pulsantiera che controlla i movimenti di porte e finestre in un modello di casa robotizzata. Quest'ultima è dotata di un sistema antifurto che scatta quando viene aperta una finestra, mentre le porte possono essere aperte solo dopo aver digitato sul computer una parola d'ordine. I due programmi di controllo vengono eseguiti contemporaneamente, grazie all'elaborazione parallela consentita dal C-LOGO.
I tre piani della casa sono collegati da un ascensore, programmato per fermarsi ad ogni piano (il computer deve sapere a quale piano si trova l'ascensore per poterlo azionare correttamente). Costruito con una scatola di montaggio "Meccano", l'ascensore viene mosso da un motore a corrente continua, collegato a tre interruttori a lamelle, che indicano il piano a cui l'ascensore si trova. Il programma sposta l'ascensore al piano terra, al primo o al secondo piano quando sulla pulsantiera vengono premuti i rispettivi pulsanti di controllo. Inizialmente gli studenti hanno ritenuto opportuno far ignorare all'ascensore eventuali pressioni dei pulsanti durante il suo movimento. In seguito, però, hanno deciso che la soluzione migliore è far registrare al computer i segnali in arrivo per poi eseguirli in sequenza.

Soluzioni semplici

Grazie alle caratteristiche del C-LOGO, i ragazzi possono risolvere con semplicità problemi che altrimenti richiederebbero una complessa programmazione. Per far funzionare l'ascensore, ad esempio, nel programma sono state introdotte tre routine con funzioni di rivelatore di segnali, di sequenziatore dei comandi e di controllo sull'ascensore. La prima e la terza ricorrono alla elaborazione parallela. I segnali, intercettati dalla funzione FOREVER, vengono passati al sequenziatore, che li memorizza nella corretta sequenza. Quindi l'unità di controllo chiede al sequenziatore la destinazione dell'ascensore, spostandolo poi al piano indicato e tornando a chiedere la destinazione successiva.
Molti studiosi di pedagogia ritengono che un precoce approccio alla programmazione dei robot favorisca, tra l'altro, il superamento di alcuni limiti, tipici dei processi mentali infantili. L'esperienza di tipo empirico e il conseguente sviluppo di un ragionamento più articolato permettono al bambino di raggiungere con maggior rapidità una visione più realistica del mondo circostante, basata su una continua revisione e correzione dei propri giudizi. Le attuali potenzialità didattiche dei robot sono paragonabili a quelle offerte all'inizio degli anni '80 dai computer. La comparsa di linguaggi come il LOGO e il C-LOGO, il calo nei prezzi dell'hardware e il crescente sviluppo commerciale della robotica aprono un brillante avvenire all'impiego dei robot come strumenti educativi.

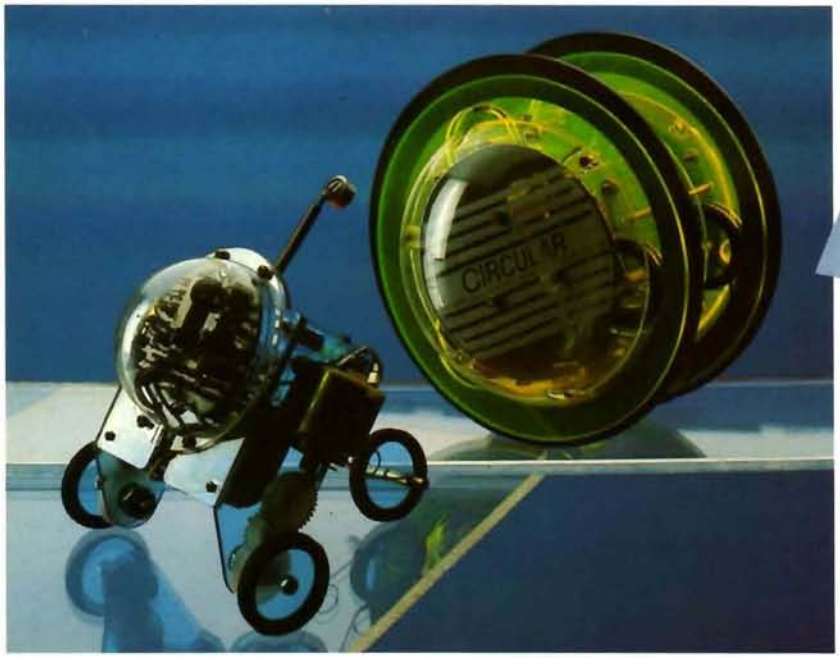
IMPARARE CON LA TARTARUGA

L'uso della tartaruga nelle aule scolastiche non è limitato alla geometria, alla trigonometria e alla programmazione. Alcuni progetti didattici implicano la costruzione di labirinti (quello mitologico di Teseo e del Minotauro oppure quello rappresentato nel pavimento della cattedrale di Rheims). Un progetto particolarmente ambizioso riguarda la costruzione di un modellino di città che la tartaruga deve attraversare, fermandosi in punti prestabiliti per acquistare gli oggetti elencati in una lista.
Esistono anche giochi didattici per la tartaruga, quali "Racetrack" e "Postman": nel primo gli allievi devono completare la programmazione della tartaruga per farle eseguire un dato percorso; nel secondo i bambini si dispongono in cerchio e si inviano messaggi scritti, trasportati dalla tartaruga.
Altri due giochi, dedicati all'insegnamento della geometria, sono "Turnturtle" e "Shoveturtle". Nel primo la tartaruga viene posta al centro di un cerchio diviso in settori numerati: l'allievo deve programmarla a ruotare di un dato angolo e a puntare verso un particolare settore. Vince il bambino che ottiene il punteggio massimo col minor numero di comandi. Il gioco "Shoveturtle" segue lo stesso principio, ma la tartaruga si muove lungo una linea retta divisa in segmenti



Sulle tracce del Big Trak

Questo veicolo semovente può essere programmato, secondo principi affini a quelli del LOGO, tramite la sua tastierina numerica. I comandi previsti sono composti da un'istruzione di direzione (destra, sinistra, ecc.) seguita dal numero dei passi (da 1 a 99). Il Big Trak memorizza ed esegue in sequenza fino ad otto diversi comandi





Movimento rotatorio

Il Circular non presenta difficoltà di costruzione. Le schede dei circuiti stampati e la scatola del radiocomando arrivano assemblate, quindi la realizzazione non richiede operazioni di saldatura né specifiche conoscenze in campo elettronico

Piccoli robot

Movits è una serie di economici piccoli apparecchi assimilabili a robot

È opinione diffusa che la robotica sia destinata a divenire la base della futura abitazione computerizzata. L'attuale livello di sviluppo di questo settore è paragonabile a quello raggiunto dai computer agli inizi degli anni '70. La comparsa di tecnologie sempre più avanzate, inoltre, favorisce l'ideazione di prodotti completamente nuovi. Mentre all'inizio i micro si sono diffusi faticosamente e solo grazie all'impegno di una ristretta cerchia di appassionati d'elettronica, l'attuale rivoluzione microelettronica viene seguita con interesse dal grande pubblico.

Non deve quindi meravigliare la comparsa sul mercato di una serie di robot, relativamente economici, destinati ad impieghi domestici (definibili, per analogia, "home robot"). Gli apparecchi della serie Movits, fabbricati in Giappone dalla Kaho Musen Company, rappresentano uno dei primi tentativi di produzione su larga scala in questo settore. Venduti in scatola di montaggio, essi non richiedono l'acquisto di componenti extra, fatta eccezione per le pile da 1,5 volt (addirittura, ogni kit contiene il cacciavite necessario al montaggio). Rivestiti da una robusta custodia plastica, trasparente nella parte superiore per consentire la visione dei circuiti interni, i Movits hanno un aspetto decisamente fantascientifico. La serie è composta da 5 modelli. Esaminiamo per primo quello più economico: il Monkey, un robot controllato tramite segnali sonori. Quando l'utente emette un grido o batte le mani, il sensore di suono capta il segnale e aziona alternativamente i due bracci dell'apparecchio che muovono il robot lungo un cavo o una guida d'altro tipo. Il sensore è un microfono a condensatore, collegato ad un circuito che decodifica i segnali. Un apposito segnale avvia il motore elettrico a corrente continua, che a sua volta aziona un meccanismo a gomito per muovere i bracci meccanici. Un orologio interno consente di predeterminare il momento dell'arresto del Monkey.

Il secondo robot della serie, il Tracer II, è controllato da un sensore a raggi infrarossi applicato alla sua base, che permette all'apparecchio di seguire una linea tracciata sul pavimento. Viene azionato da una coppia di motori a corrente continua, che azionano le sue tre ruote. Di prezzo leggermente superiore è il Piper Mouse, guidato da un sensore che capta le emissioni di un fischietto ad ultrasuoni (quelli usati per i cani). In risposta ai segnali ricevuti, il robot esegue una serie di comandi preselezionati: gira a sinistra, si ferma, gira a destra, si ferma, avanza, si ferma. Con questa sequenza il Piper Mouse esegue un movimento rettilineo.

Il Circular Movit è l'unico apparecchio totalmente radio-comandato della serie. Possiede due motori a corrente continua che muovono due larghe ruote applicate sul margine esterno delle cupole di rivestimento. I due pulsanti sul box di controllo pilotano i due motori: premendoli entrambi, il robot si muove in avanti; agendo su un solo pulsante, l'apparecchio ruota nella direzione corrispondente (a destra o a sinistra). La rotazione indipendente dei motori è controllata da due schede elettroniche, che ricevono i differenti segnali radio emessi dalla scatola di controllo.

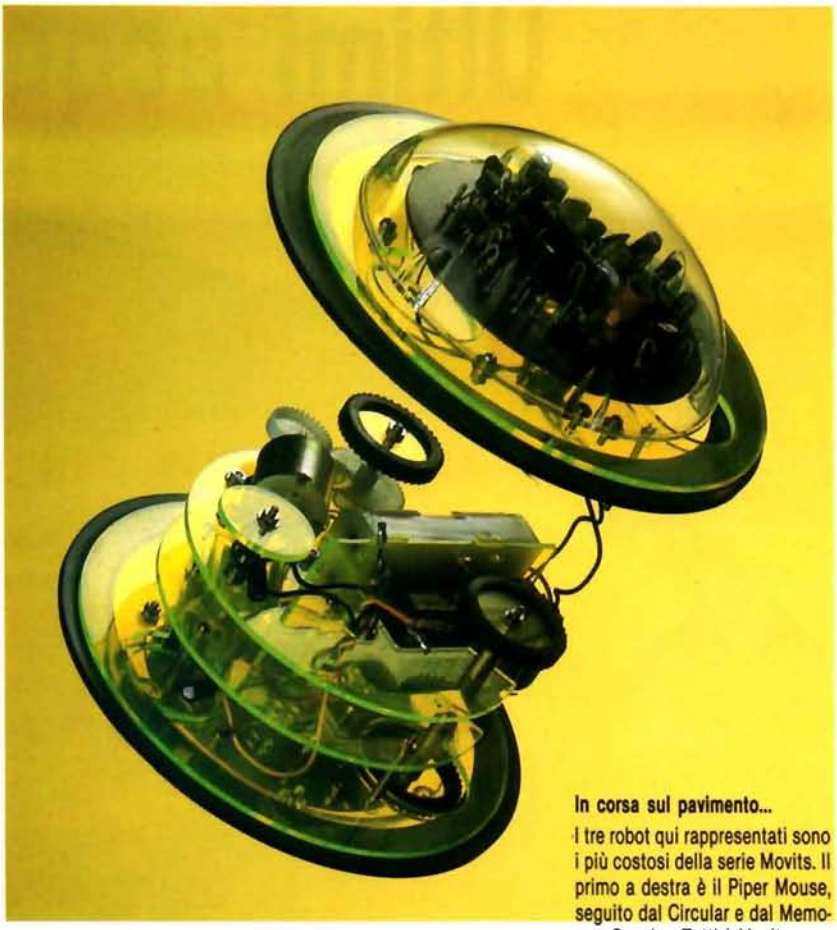
È un robot?

L'assemblaggio delle macchine appena descritte è semplice e divertente mentre è piuttosto complicato descriverne l'architettura interna. Il robot più "avanzato" è il Circular, ma il suo sistema di controllo non è nuovo: viene infatti usato da tempo sui modelli radiocomandati di aerei e navi. Per essere

considerato un vero e proprio robot, un simile dispositivo radiocomandato dovrebbe possedere un sistema di controllo interno programmabile, capace di far eseguire all'apparecchio una sequenza di movimenti. Il Memocon Crawler è una macchina che si avvicina molto alle specifiche particolarità di un robot e, come gli altri Movits, viene azionato da motori a corrente continua. Il sistema di guida si basa su una tastiera collegata al Crawler da un cavo a nastro. La tastiera possiede 5 diversi interruttori, relativi ai 5 comandi previsti (avanti, destra, sinistra, arresto, emissione di un segnale acustico) al cui azionamento corrisponde l'accensione di diodi LED sul robot. I cinque comandi sono memorizzati in una RAM statica con una memoria di 256 byte (ma qui ciascun byte è composto da 4 bit). I byte nei quali sono registrati i comandi non possono essere indirizzati individualmente, ma solo in modo sequenziale. Non è quindi possibile ripetere singolarmente i movimenti della sequenza se non rilanciandola per intero.

I robot Movits, come già accennato, sono venduti in scatola di montaggio. Tuttavia il loro assemblaggio non è così complicato come il gran numero di componenti potrebbe far pensare: anche un bambino può eseguirlo. D'altra parte, è stato proprio il figlio decenne del presidente della Kaho Musen Company il primo a montare i modelli dimostrativi dei Movits. I diversi pezzi dei robot sono imballati in contenitori numerati. Le istruzioni per il montaggio comprendono un completo schema d'assemblaggio del robot, sul quale sono rappresentati graficamente i diversi pezzi, indicati dal numero del relativo contenitore. Le frecce dello schema indicano l'esatta collocazione di ogni componente. Il manuale di assemblaggio contiene anche una serie di istruzioni scritte, per facilitare il riconoscimento dei pezzi che potrebbero essere confusi. Poiché la scheda del circuito stampato è già assemblata, i soli attrezzi necessari per il montaggio sono una pinza, un cacciavite e un piccolo martello.

Nonostante il loro aspetto futuristico, i robot della serie Movits devono essere considerati più sofisticati giocattoli elettronici che non veri e propri robot. Grazie al loro prezzo contenuto, però, rappresentano un'ottima base per avvicinare il pubblico (soprattutto quello dei bambini), all'affascinante mondo della robotica.



In corsa sul pavimento...
I tre robot qui rappresentati sono i più costosi della serie Movits. Il primo a destra è il Piper Mouse, seguito dal Circular e dal Memocon Crawler. Tutti i Movits possiedono un rivestimento trasparente, che permette di osservare i motori e le componenti elettroniche e che dà agli apparecchi un aspetto futuristico



CHRIS STEVENS

MONKEY
Movimento
due bracci prensili azionati alternativamente da un ingranaggio
Controllo
sensore sonoro
Alimentazione
due pile da 1,5 volt

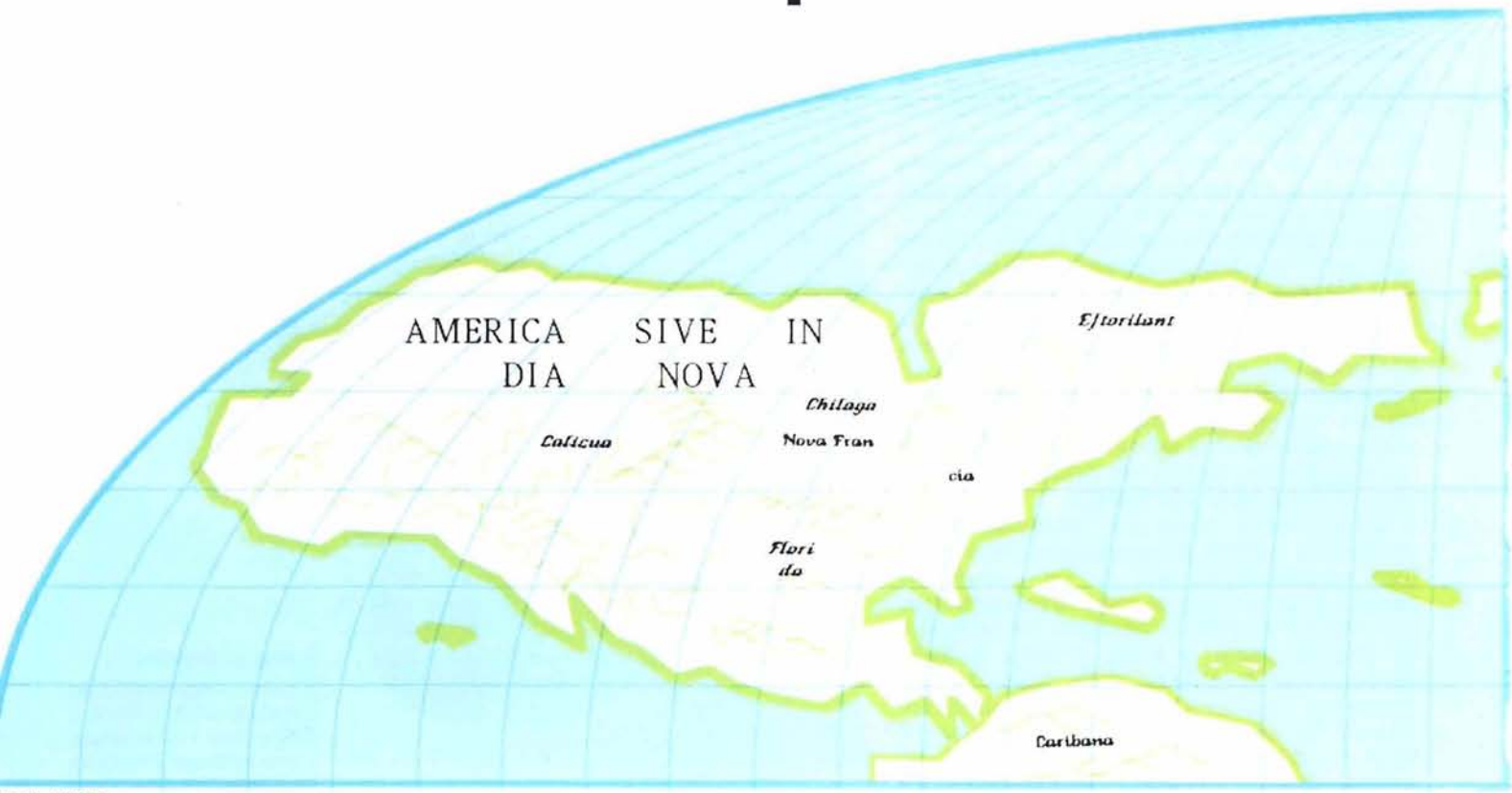
LINE TRACER II
Movimento
tre ruote azionate da due motori a c.c.
Controllo
sensore infrarosso
Alimentazione
due pile da 1,5 volt e una da 9 volt

PIPER MOUSE
Movimento
tre ruote mosse da due motori a c.c.
Controllo
sensore ad ultrasuoni
Alimentazione
due pile da 1,5 volt e una da 9 volt

CIRCULAR
Movimento
due ruote con demoltiplica dentata, azionate da due motori a c.c.
Controllo
scatola di radiocomando
Alimentazione
tre pile da 1,5 volt e due da 9 volt

MEMOCON CRAWLER
Movimento
tre ruote mosse da due motori elettrici a c.c.
Controllo
memoria programmabile
Alimentazione
due pile da 1,5 volt e una da 9 volt

Ultimi acquisti



IAN McKINNELL

Siamo quasi pronti a salpare: assicuriamoci di avere tutto l'occorrente

Il Nuovo Mondo

I viaggi che hanno portato a grandi scoperte geografiche sono avvenuti intorno alla fine del XV secolo. Nel XVI secolo gli esploratori avevano ancora idee molto vaghe sulla conformazione dei continenti della Terra: le loro mappe erano appena abbozzate e le isole, o addirittura continenti interi, erano disposti in modo errato. Il gioco "New World" si sviluppa su una di queste mappe antiche, sulle quali i continenti venivano spesso rappresentati come masse collegate tra loro

L'equipaggio verrà pagato quando (e se!) la nave farà ritorno al porto. Il giocatore deve decidere quanto investire in merci di scambio, sperando di trarne un buon profitto, e quanto denaro mettere da parte (se la cassa non è già vuota), per pagare l'equipaggio al ritorno. Le merci disponibili sono: medicinali, armi da fuoco, sacchi di sale, pezze di stoffa, coltelli e bigiotteria. Il programma comunica i prezzi delle merci, chiede al giocatore le quantità che desidera acquistare e, dopo ogni transazione, visualizza la situazione di cassa. Se il giocatore non ha denaro a sufficienza per pagare un ordine, il programma invita a modificarlo. La stiva della nave è talmente ampia, da poter contenere qualsiasi quantità di merce. Ultimati gli acquisti, il viaggio ha inizio. Prima che la nave salpi, comunque, sullo schermo compare un elenco dei membri dell'equipaggio, delle provviste e delle merci acquistate e la situazione di cassa aggiornata.

Come nelle prime due fasi di preparazione, all'inizio del programma vengono dimensionate le matrici per la terza fase, riferite ai nomi, ai prezzi e alle quantità di beni acquistati. Queste matrici vengono aggiunte alle precedenti istruzioni d'inizializzazione a partire dalla linea 30. I codici delle diverse merci acquistate vengono inseriti nella matrice OA( ), mentre i loro nomi nella matrice D$( ), dimensionata alla linea 31. I sei elementi da D$(1) a D$(6) della matrice vengono definiti nelle due linee successive e contengono rispettivamente i nomi: BOTTIGLIE DI MEDICINALI, ARMI, SACCHI DI SALE, PEZZE DI STOFFA, COLTELLI e MONILI. Come già detto, l'impiego degli indirizzi della matrice, anziché dei nomi delle merci, facilita la programmazione del gioco.

La matrice OC( ) a sei elementi, che contiene i prezzi delle merci, viene dimensionata e inizializzata alle linee 34 e 35. Il primo elemento OC(1) rappresenta il prezzo di ciascuna bottiglia di medicinale: una moneta d'oro. Le armi costano 5 monete d'oro (OC(2) = 5), cinque sacchi di sale 1 moneta (OC(3) = 0.2), una pezza di stoffa 2 monete (OC(4) = 2), un coltello mezza moneta (OC(5) = 0.5) e i monili 1 moneta d'oro ciascuno (OC(6) = 1).

Nel programma principale, la linea 500 chiama la routine per l'assunzione dell'equipaggio, la 550 la subroutine di acquisto delle provviste. Occorre adesso inserire la linea 600, che chiama la subroutine per l'acquisto delle merci di scambio (linee 3000–3999). Dopo aver cancellato lo schermo e introdotto una pausa, la routine visualizza e sottolinea il titolo della terza fase (linee 3005 e 3010),

avvertendo il giocatore che può aver bisogno di altre merci per il viaggio.
La routine di acquisto delle merci è contenuta in un ciclo, ripetuto una volta per ogni prodotto, e usa T come contatore di ciclo da 1 a 6. Essa comunica al giocatore il costo delle singole merci e il suo nome, estraendolo dalla matrice D$(T), alla linea 3070. Durante il primo passaggio nel ciclo, che corrisponde all'articolo BOTTIGLIE DI MEDICINALE, T viene posto ad 1. Il suo prezzo unitario viene indicato da OC(T) alla linea 3080. Con T a 1, OC(T) rappresenta una moneta d'oro, ossia il costo di una bottiglia. Al secondo passaggio, T viene incrementato a 2, D$(T) contiene l'articolo ARMI e OC(T) il prezzo unitario 5. La procedura si ripete finché T non raggiunge il valore 6.

Redazione dei messaggi

Dopo una pausa, la linea 3095 visualizza il messaggio VUOI ACQUISTARE QUALCOSA (S/N)? e la 3110 attende la risposta, inserendola poi nella variabile P$. La linea 3120 esamina il primo carattere a sinistra di P$. Se questo è N, il controllo torna all'inizio del ciclo e la routine passa all'articolo merceologico successivo. Se il primo carattere è S (per Sì), dopo una breve pausa compare il messaggio QUANTITÀ DELL'ACQUISTO? (linea 3130). La linea 3135 attende la risposta che può essere un numero, seguito o meno dal nome della merce (ad esempio, 10 oppure 10 SACCHI DI SALE). Questa risposta è valida perché la variabile VAL alla linea 3140 legge i caratteri numerici all'inizio della frase, ignorando del tutto il testo.
Eseguita l'ordinazione, la linea 45 verifica se il costo unitario dell'articolo OC(T), moltiplicato per la quantità ordinata TT è superiore alle disponibilità di cassa MO. In tal caso, il controllo passa alla linea 3150, che comunica al giocatore l'insufficienza dei suoi fondi. Dopo una pausa, la linea 3154 visualizza il messaggio PROVA DI NUOVO. Il controllo torna quindi alla linea 3130, per una diversa ordinazione. Se stavolta il denaro è sufficiente, il programma salta alla linea 3160 che sottrae la somma richiesta dal totale di cassa applicando la seguente formula: MO=MO−(OC(T)*TT). MO adesso rappresenta il denaro rimasto. Il costo dell'articolo, ossia OC(T) è moltiplicato per la quantità acquistata (TT) e il risultato è sottratto dall'ammontare della cassa. TT viene registrato nella matrice OA( ) alla linea 3165 e, dopo una pausa, la linea 3176 visualizza la quantità di monete d'oro rimaste. Dopo un'altra pausa, T viene incrementato di 1 e il controllo torna dalla linea 3200 all'inizio del ciclo, per l'acquisto di nuova merce.
Terminato l'acquisto delle merci, la linea 3210 comunica al giocatore la conclusione di questa fase di gioco. La linea 3230 lo invita a premere un qualsiasi tasto per continuare, usando la variabile K$ definita all'inizio del programma, prima di restituire il controllo al programma principale. Infine, la routine Ricapitolazione d'apertura visualizza una lista delle merci ordinate e dei membri dell'equipaggio, prima di dare inizio al viaggio.
La linea 645 usa un ciclo per visualizzare i membri dell'equipaggio appartenenti alle cinque categorie previste, mentre la linea 650 verifica la matrice CC(T) per scoprire se tutte le categorie fanno parte dell'equipaggio. Se il contatore T è a 0, la linea 670 riporta il controllo all'inizio del ciclo e incrementa T di 1. Se l'equipaggio comprende tutti i tipi, ne viene visualizzato il numero e la categoria. La lista delle scorte viene realizzata allo stesso modo dalle linee 675-710. Anche in questo caso la routine usa un ciclo per i 4 tipi di scorte previste.

Considerazioni particolari

La lista delle merci di scambio viene invece compilata con un criterio diverso, poiché la routine tratta separatamente i diversi articoli. Se una data merce non è stata acquistata, il programma passa direttamente all'articolo successivo. La linea 730, ad esempio, controlla se il giocatore ha comprato medicinali: in

Merci di scambio

Queste sono le merci che gli esploratori del XVI secolo acquistavano e caricavano sui loro velieri per commerciare con gli abitanti del Nuovo Mondo. Nella nostra simulazione, i nomi dei diversi articoli vengono registrati negli elementi della matrice D$( ), il loro prezzo unitario nella matrice OC( ) e le quantità acquistate nella matrice OA( )



IAN McKINNELL

caso negativo, si passa direttamente all'acquisto successivo (linea 740), altrimenti è visualizzata la quantità di medicinali comprati.

Quando la lista è completa, la linea 792 comunica al giocatore le rimanenze di cassa e la linea 797 visualizza il messaggio PREMI UN TASTO PER INIZIARE IL VIAGGIO.

MODULO TRE: MERCI DI SCAMBIO

Dimensionamento delle matrici

```
30 DIM OA(6)
31 DIM D$(6)
32 D$(1)="BOTTIGLIE DI MEDICINALE": D$(2)="ARMI":
   D$(3)="SACCHI DI SALE"
33 D$(4)="PEZZE DI STOFFA": D$(5)="COLTELLI": D$(6)="MONILI"
34 DIM OC(6)
35 OC(1)=1:OC(2)=5:OC(3)=.2
36 OC(4)=2:OC(5)=.5:OC(6)=1
```

Chiamata alla subroutine per acquisto delle merci

```
600 GOSUB 3000: REM ACQUISTO MERCI DI SCAMBIO
```

Subroutine di ricapitolazione d'apertura

```
605 REM **** PRONTI A PARTIRE ****
610 PRINT CHR$(147)
615 S$="SEI PRONTO A INIZIARE*": GOSUB 9100
625 S$=" IL VIAGGIO.*": GOSUB 9100
630 GOSUB 9200
635 PRINT: S$="COMPOSIZIONE DELL'EQUIPAGGIO:*":GOSUB 9100
640 GOSUB 9200
645 FOR T=1 TO 5
650 IF CC(T)=0 THEN 670
655 PRINT C$(T);": ";
660 PRINT TAB(15);CC(T)
668 GOSUB 9200
670 NEXT
674 GOSUB 9200
675 PRINT: S$="HAI LE SEGUENTI SCORTE:*":GOSUB 9100
680 GOSUB 9200
685 FOR T=1 TO 4
690 IF PA(T)=0 THEN 710
695 PRINT PA(T);U$(T); " DI ";
700 PRINT P$(T)
708 GOSUB 9200
710 NEXT
715 GOSUB 9200
720 PRINT: S$="HAI INOLTRE ACQUISTATO:*": GOSUB 9100
725 GOSUB 9200
733 IF OA(1)=1 THEN S$="BOTTIGLIA DI MEDICINALE *":GOTO 735
734 S$="BOTTIGLIE DI MEDICINALE*"
735 PRINT OA(1);TAB(8);:GOSUB 9100
736 GOSUB 9200
740 IF OA(2)=0 THEN 750
743 IF OA(2)=1 THEN S$="ARMA DA FUOCO*": GOTO 745
744 S$="ARMI DA FUOCO*"
745 PRINT OA(2);TAB(8);:GOSUB 9100
746 GOSUB 9200
750 IF OA(3)=0 THEN 760
753 IF OA(3)=1 THEN S$="SACCO DI SALE*": GOTO 755
754 S$="SACCHI DI SALE*"
755 PRINT OA(3);TAB(8);:GOSUB 9100
756 GOSUB 9200
760 IF OA(4)=0 THEN 770
763 IF OA(4)=1 THEN S$="PEZZA DI STOFFA*": GOTO 765
764 S$="PEZZE DI STOFFA*"
765 PRINT OA(4);TAB(8);:GOSUB 9100
766 GOSUB 9200
770 IF OA(5)=0 THEN 780
773 IF OA(5)=1 THEN S$="COLTELLO*": GOTO 775
774 S$="COLTELLI*"
775 PRINT OA(5);TAB(8);:GOSUB 9100
776 GOSUB 9200
780 IF OA(6)=0 THEN 790
783 IF OA(6)=1 THEN S$="MONILE*": GOTO 785
784 S$="MONILI*"
785 PRINT OA(6);TAB(8);:GOSUB 9100
786 GOSUB 9200
790 GOSUB 9200
792 PRINT: PRINT "RESTANO ";MO;" MONETE D'ORO"
796 GOSUB 9200
797 S$="PREMI UN TASTO:INIZIA IL VIAGGIO*"
798 GOSUB 9100
799 GET P$:IF P$=""THEN 799
```

Subroutine "Merci di scambio"

```
3000 REM **** FASE 3: MERCI DI SCAMBIO ****
3001 PRINT CHR$(147): REM FASE 3
3002 GOSUB 9200
3005 PRINT "     FASE 3 - ALTRE MERCI  "
3010 PRINT "     ------   -----------  "
3020 GOSUB 9200
3025 PRINT
3030 S$="DURANTE IL  VIAGGIO, POSSONO*": GOSUB 9100
3035 S$="OCCORRERE ALTRE COSE, AD ESEMPIO*":
     GOSUB 9100
3040 S$="MEDICINALI, MERCI DI SCAMBIO,*": GOSUB 9100
3046 GOSUB 9200
3050 S$="OPPURE... ARMI.*":GOSUB 9100
3055 GOSUB 9200:GOSUB 9200
3060 FOR T=1 TO 6
3065 PRINT
3070 PRINT D$(T);":"
3075 S$="COSTO IN MONETE D'ORO:*": GOSUB 9100
3080 PRINT OC(T)
3090 GOSUB 9200
3095 S$="VUOI ACQUISTARNE (S/N)*": GOSUB 9100
3110 GET P$ : IF P$= "" THEN 3110
3115 IF P$()"S" AND P$()"N" THEN 3095
3120 IF P$="N" THEN 3175
3125 GOSUB 9200
3130 S$="QUANTITA' DELL'ACQUISTO?*": GOSUB 9100
3135 INPUT P$
3140 TT=VAL(P$)
3145 IF OC(T) * TT ) MO THEN 3150
3147 GOTO 3160
3150 S$="DENARO INSUFFICIENTE!!*":GOSUB 9100
3152 GOSUB 9200
3154 S$="DEVI MODIFICARE LA QUANTITA'...*":GOSUB 9100
3155 GOSUB 9200:GOTO 3130
3160 MO=MO-(OC(T)*TT)
3165 OA(T)=TT
3170 GOSUB 9200
3175 PRINT
3176 PRINT" MONETE RIMASTE = ";MO
3200 GOSUB 9200:NEXT T
3205 GOSUB 9200:PRINT:PRINT
3210 S$="FINE DELLA FASE 3*": GOSUB 9100
3220 GOSUB 9200:PRINT
3230 S$=K$:GOSUB 9100
3240 GET P$:IF P$="" THEN 3240
3999 RETURN
```

Basic & Basic

Spectrum:

Sostituire le seguenti linee:

```
 32 DIM (6,20)
620 CLS
799 LET   P$ = INKEY$:IF
    P$ = "" THEN GO TO
    799
3001 CLS
3110 INPUT P$:LET P$(1 TO 1)
3240 LET  P$ = INEKEY$:IF
    P$ = "" THEN GO TO
    3240
```

Micro BBC:

Sostituire le seguenti linee:

```
610 CLS
799 A$ = GET$
3001 CLS
3240 A$ = GET$
```

DATABASE

Ecco l'ultima tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi

DESCRIZIONE	INDIRIZZO
String concatenation concatenazione di stringhe	9C15
Swap FAC#1 and (&4B) scambia FAC#1 con (&4B)	A4D6
TAB(X routine funzione TAB(X	8E24
TAB(X) routine routine funzione TAB(X)	8E40
TAN function routine routine funzione TAN	A6BE
TIME function routine routine funzione TIME	AEB4
TIME statement routine routine istruzione TIME	92C9
TOP function routine routine funzione TOP	AEDC
TRACE statement routine routine istruzione TRACE	9295
TRUE function routine routine funzione TRUE	ACC4
Token table tabella dei 'token'	8071
Tokenise a line of text 'tokenizza' una linea di testo	8951
Tokenise line numbers 'tokenizza' i numeri di linea	88DB
UNTIL statement routine routine istruzione UNTIL	BBB1
USR function routine routine funzione USR	ABD2
Unpack FAC#1 from (&4B) decompatta FAC#1 da (&4B)	A3B2
Unpack FAC#2 from (&4B) decompatta FAC#2 da (&4B)	A34E
Unstack a parameter estrae un parametro dallo stack	8CC1
VAL function routine routine funzione VAL	AC2F
VDU statement routine routine istruzione VDU	942F
VPOS function routine routine funzione VPOS	AB76
WIDTH statement routine routine istruzione WIDTH	B4A0
Warm start avvio sistema 'a caldo'	8AF3
Zero FAC#1 azzera FAC#1	A686
Zero FAC#2 azzera FAC#2	A453
^ operator routine routine operatore ^	9E35

INDIRIZZO	DESCRIZIONE
BF78	**OPENIN function routine** routine funzione OPENIN
BF7C	**OPENOUT function routine** routine funzione OPENOUT
BF80	**OPENUP function routine** routine funzione OPENUP
BF99	**CLOSE statement routine** routine istruzione CLOSE
BFA9	**Get file handle** routine gestione file
BFCF	**Print a string** visualizza una stringa
BFE4	**REPORT statement routine** routine istruzione REPORT

NOTA. La colonna che riporta gli indirizzi HEX in ordine crescente è leggermente più breve dell'altra in quanto non esistono ripetizioni.

Hybrid Music 500

Un interessante accessorio musicale per il Micro BBC

Il Music 500, benché commercializzato dalla Acorn per il Micro BBC, è stato progettato da un'altra compagnia di Cambridge (GB), la Hybrid Technology. Il Music 500 è un sofisticato apparecchio elettronico, racchiuso in un contenitore color crema simile a quello delle unità a dischetti. Tramite la sua presa DIN a 5 poli, è collegabile al connettore del bus da 1 MHz del Micro BBC e ad un amplificatore stereofonico. All'interno della scatola trovano posto 16 oscillatori, programmabili individualmente in frequenza, volume, bilanciamento e forma d'onda, attraverso il computer principale. Grazie al software di corredo, la qualità del Music 500 supera quella di numerosi sintetizzatori molto più costosi.

Esistono due metodi per sintetizzare i suoni. Gli apparecchi più economici generano una semplice onda sinusoidale che viene poi distorta e filtrata fino ad ottenere l'effetto desiderato.

Sebbene questa tecnica consenta di produrre una discreta varietà di suoni, anche nelle mani di un esperto risulta abbastanza limitata. Alcuni sintetizzatori più costosi adoperano invece alcune onde sinusoidali, l'una modulata dall'altra. In altre parole usano l'ampiezza di un'onda per controllare la frequenza di un'altra. Questo metodo di controllo ha una maggiore efficacia rispetto alla semplice distorsione, ma non consente di riprodurre con precisione ogni suono "naturale" (consultare le pagg. 581-584).

Il Music 500 adotta un terzo metodo, completamente diverso: crea le proprie forme d'onda con una tecnica digitale, secondo una configurazione programmata, campionandole 47.000 volte al secondo. Oltre alla capacità di programmare qualsiasi tipo di onda, il Music 500 offre numerose tecniche di modulazione (non solo quella di frequenza) e può combinare le emissioni sui canali con la modulazione ad anello e la modulazione di fase. Infine, può usare forme d'onda diverse dalle semplici onde sinusoidali.

Il software di corredo, che semplifica notevolmente le creazioni musicali sull'apparecchio, è fornito su cassetta, ma può essere facilmente copiato su disco usando un apposito programma. La Hybrid Technology ha annunciato il lancio, entro breve termine, di una versione estesa del software, conservato su ROM. Il software di corredo comprende l'AMPLE, un linguaggio di programmazione musicale che permette di sfruttare al meglio le potenzialità del Music 500. L'AMPLE, che per alcuni versi ricorda il FORTH e il LOGO, è un linguaggio basato su "parole", che consentono al programmatore di definire funzioni e procedure. Una tipica composizione musicale in AMPLE potrebbe essere così strutturato:

```
10 .%IL VOLO DEL CALABRONE
20 .preparazione
30 .esecuzione
```

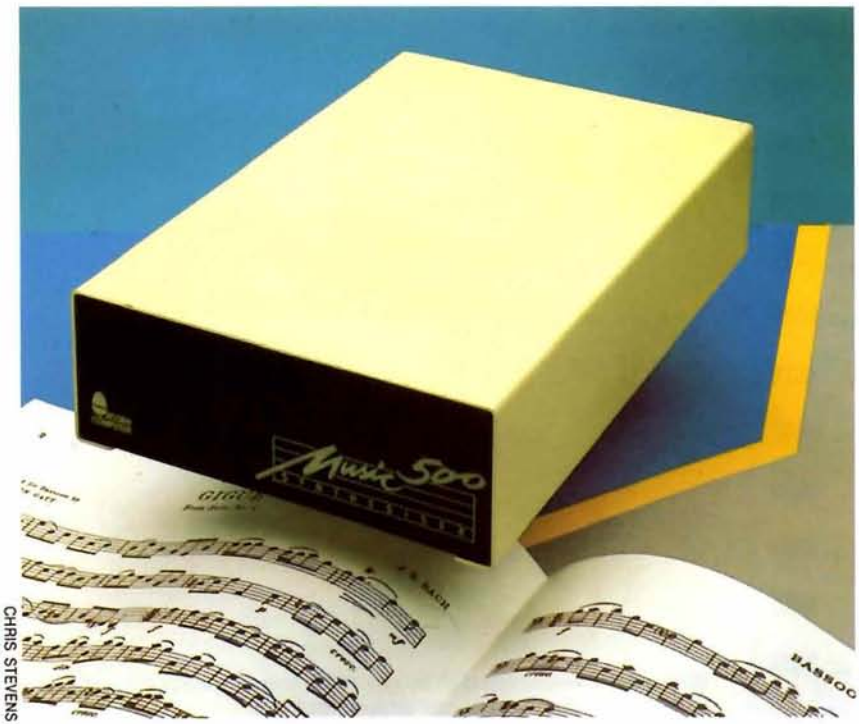
Il termine 'preparazione' è una parola definita dall'utente, programmata in precedenza nel seguente modo:

```
10 .'preparazione'
20 .[
30  .prepsint
40 .prepbasso
50 .preppercussioni
60 .preppiatti
70 .]
```

Gli elementi della seconda procedura sono stati a loro volta pre-programmati per generare diversi suoni (quello del basso o quello dei piatti, ad esempio). Il Music 500 dispone di forme d'onda predefinite, che consentono un approccio semplificato alla programmazione dell'apparecchio. Comunque, la creazione di nuove forme d'onda non è difficile. Queste possono essere definite intervenendo sulle proporzioni relative delle prime 16 armoniche oppure con una tecnica di selezione di tipo matematico. Questo secondo metodo permette al Music 500 di generare un "rumore" definito da una forma d'onda casuale. Ad

"Byte" in musica

L'Hybrid Music 500 è indubbiamente una delle più sofisticate periferiche musicali progettate per home computer. Dotato di 16 oscillatori e di una modulazione d'onda digitale, questo strumento è un pericoloso concorrente dei sintetizzatori di prezzo molto più elevato



CHRIS STEVENS

esempio, la procedura per il suono dei piatti potrebbe avere la seguente struttura:

```
10 .'piatti'
20 .[
30 .3WMOD 0 rand! 128 FOR (RAND? INDEX WG!) FOR WGC
40 .1 CHAN
50 .3 WAVE ON RM —1 POS
60 .2 CHAN
70 .3 WAVE 2000 OFFSET 1 POS
80 .]
```

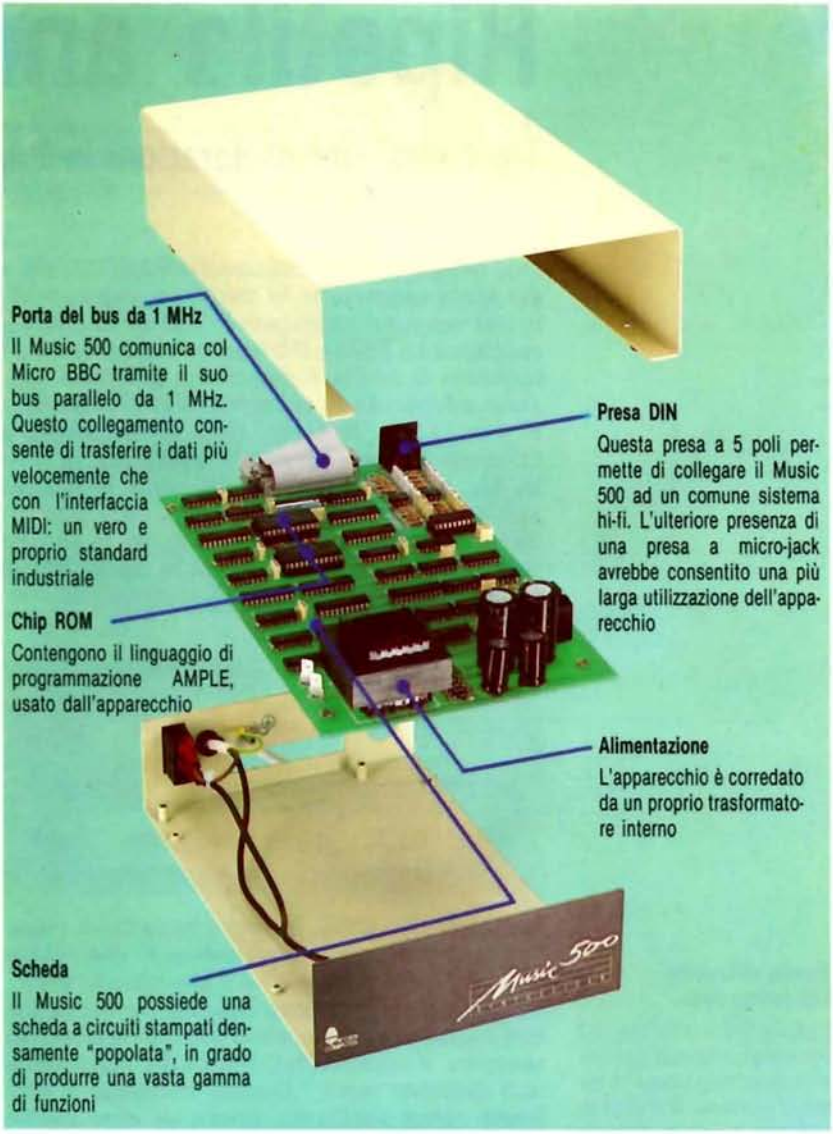
L'equivalente AMPLE del FOR...NEXT del BASIC assegna a ciascun punto dell'onda un valore casuale. Quest'onda viene usata per i due canali che emettono il suono (oscillatori): l'altezza del secondo viene sfalsata rispetto al primo, assume una diversa posizione stereofonica e viene inviata su un altro canale stereo. Il risultato è un suono esplosivo e fragoroso come quello dei piatti.
In alternativa, è possibile definire una forma d'onda di tonalità più gradevole:

```
10 .'sint'
20 .[
30 .WZERO 181 1 WH! 135 2 WH! 181 3 WH! 62 4 WH! 76 5 WH!
   37 6 WH! 14 7 WH! 3 8 WH!
40 .WHG WGC 3
50 .]
```

Questo tipo d'onda viene costruito variando le proporzioni delle prime 8 armoniche. Gli inviluppi dell'altezza e dell'ampiezza vengono creati in modo simile, ricorrendo alle semplici forme d'onda predefinite all'interno dell'apparecchio o a quelle create dall'utente. Per richiamare i diversi suoni programmati, facendoli eseguire alle varie "voci", si potrebbe usare una parola del tipo "esecuzione"

```
10 ."esecuzione"
20 .[
30 .4 PLAYERS
40 .1 PLAY (melodia del pianoforte) PLAY
50 .2 PLAY (accompagnamento del basso) PLAY
60 .3 PLAY (base ritmica delle percussioni) PLAY
70 .4 PLAY (suono dei piatti) PLAY
80 .GO
90 .]
```

Questa procedura comunica al sistema il numero degli "esecutori" del pezzo e assegna a ciascuno di essi un suono e una parte (ovviamente definite dall'utente). Infine le 4 parti vengono eseguite in accordo tra loro per produrre il brano musicale. Questa capacità "multi-operativa" dell'AMPLE permette di suonare sul Music 500 brani simili a quelli di un'orchestra, che possono essere programmati separatamente e quindi suonati assieme per ottenere un effetto polifonico. L'AMPLE è un linguaggio interamente modulare e strutturato in modo gerarchico. Persino le parole riservate possono essere ridefinite, anche se ciò implica interventi piuttosto complicati. Ad esempio, è possibile ridefinire la parola che indica la battuta in modo che la seconda nota di ogni battuta venga suonata più forte per dare maggior enfasi al brano.
Chi non mira ad ottenere risultati così sofisticati, può affidarsi ad una serie di funzioni interne al linguaggio, che consentono di suonare semplici melodie inserendo le note da tastiera senza dover definire le parole nel modo descritto in precedenza.
La Acorn ha progettato una tastiera musicale a 49 tasti, collegabile alla porta utente del Micro BBC, da abbinare al Music 500. Il software di corredo, sempre in AMPLE, dovrebbe contenere un "elaboratore di note", col quale si può registrare, riprodurre e arrangiare i brani suonati sulla tastiera.



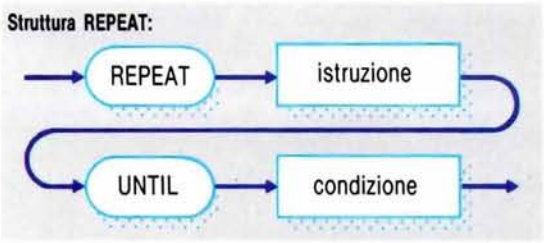
CHRIS STEVENS

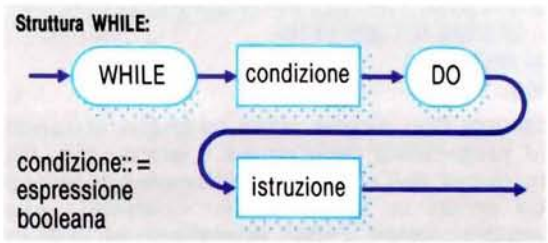
MUSIC 500	
Oscillatori	16
Gamma di frequenze	0-20 Khz
Risoluzione dell'altezza	un sedicesimo di semitono
Forme d'onda	
Armonica	16 armoniche
Geometrica	128 punti
Inviluppi (altezza/ampiezza)	17 segmenti
Lunghezza di passo	0-320 secondi
Risoluzione temporale	10 ms.
Posizioni stereo	7
Accuratezza della forma d'onda	logaritmica ad 8-bit
Velocità di campionatura	46875/sec.
Risoluzione della frequenza	0,0056 Hz
Base dei tempi	255 ms-655 ms/ciclo
Simultaneità d'esecuzione	da 1 a 9 elaborazioni

Ripetila ancora, RAM

Tre diversi tipi di iterazioni in PASCAL

Due dei tre tipi di strutture del PASCAL sono già state esaminate: le strutture sequenziali, in cui vengono impiegate le parole di apertura/chiusura BEGIN e END (vedere a pag. 1504) e i costrutti di scelta o selezione, nei quali vengono adoperate le istruzioni IF o CASE (vedere a pag. 1526). Il terzo tipo di struttura è l'iterazione, per la quale esistono, in PASCAL, tre diversi costrutti: uno è basato sull'istruzione FOR, che fornisce cicli pilotati da un contatore, gli altri due sulle parole riservate WHILE e REPEAT, in cui la ripetizione è controllata dal verificarsi di determinate condizioni.





LIZ DIXON

Nel costrutto WHILE...DO, per prima cosa viene valutata l'espressione booleana delimitata da queste due parole e, se il risultato è "vero", vengono ripetutamente eseguite le istruzioni che seguono (che possono essere espressioni semplici o complesse), finché la condizione non diviene "falsa". Questa condizione booleana viene verificata prima di ogni nuovo passaggio nel ciclo, ossia prima di eseguire nuovamente il "corpo" della WHILE.
Ovviamente, ciò implica che almeno uno degli elementi dell'espressione booleana venga in qualche modo cambiato, altrimenti il ciclo proseguirebbe all'infinito. Ad esempio:

```
WHILE MaxInt > 1 DO
   WriteLn ('Il ciclo gira!')
```

non finisce mai; più realisticamente:

```
read (prelievo);
WHILE prelievo > Saldo DO
   BEGIN
      WriteLn ('Fondi insufficienti — ripeti:');
      write ('Ammontare?');
      read (prelievo)
   END
```

Questo esempio evidenzia la potenza del costrutto WHILE: se, alla prima immissione, il prelievo è minore del saldo sul conto, il ciclo WHILE non viene eseguito. Ma in caso contrario il "corpo" viene eseguito finché la condizione imposta non diviene "falsa". A questo punto l'esecuzione ritorna nel modo sequenziale, passando all'istruzione immediatamente seguente il corpo della WHILE. La posizione della espressione booleana, all'inizio della WHILE, indica sintatticamente l'affidabilità della semantica del PASCAL. Se si vuole che il ciclo venga eseguito solo se la condizione è vera, allora ci si affidi alla WHILE.
Tuttavia, esistono occasioni in cui occorre eseguire il ciclo almeno una volta, per assegnare il valore che determina la sua conclusione. In questo caso, illustrato dal programma per la sezione aurea a pag. 1556, conviene adoperare REPEAT...UNTIL, che meglio si adatta all'algoritmo di partenza.

Si noti che in tale struttura la condizione viene verificata soltanto quando si giunge al termine del ciclo: una situazione decisamente antitetica a quella della struttura WHILE. In termini più formali: il ciclo REPEAT termina quando l'espressione booleana risulta vera, mentre il ciclo WHILE termina (se mai è iniziato) non appena la condizione diviene falsa. In tema di formalità, va fatto notare che il corpo di una REPEAT, se composto da espressioni complesse, dovrebbe essere racchiuso da apposite BEGIN...END. Tuttavia, il PASCAL può farne a meno, ritenendo sufficiente la delimitazione data dalle parole riservate REPEAT...UNTIL. Ai fini di una maggior chiarezza nei listati, comunque, conviene prendere l'abitudine di inserire sia la BEGIN che la END, anche se formalmente superflue.
Il seguente segmento di programma conta quante volte compare la lettera 'e' in una frase immessa da tastiera. La frase va conclusa con un punto, per far terminare il ciclo REPEAT ed ottenere il risultato.

```
conto := 0;
REPEAT
   read (simbolo);
   IF simbolo = 'e' THEN conto := conto + 1
UNTIL simbolo = '.';
WriteLn ('Ci sono',conto :1,'"e" nella frase.')
```

In un caso simile, si può usare indifferentemente l'uno o l'altro costrutto.
Si potrebbe scrivere, con altrettanta funzionalità:

Risposta all'esercizio sulla sezione aurea

A pagina 1556 è stato proposto un esercizio, relativo al programma sulla sezione aurea e ai numeri di Fibonacci. Si trattava di modificare il ciclo in modo da farlo terminare quando la successiva coppia di numeri di Fibonacci eccedeva il valore massimo consentito da MaxInt.
Nella rappresentazione in "complemento a due", se un valore intero supera il massimo consentito da MaxInt, il bit del segno diviene attivo, trasformando il valore in un numero negativo e facendo così "inchiodare" il programma. Tuttavia, non si può imporre una condizione del tipo:

```
UNTIL primo + secondo >
MaxInt
```

poiché non esiste un numero maggiore di MaxInt per definizione. Comunque, con una banale trasformazione, si può ottenere lo scopo voluto:

```
UNTIL secondo > MaxInt - primo
```

Adesso, l'aritmetica reale e la costante Epsilon non sono più necessarie

```
conto := 0;
read (simbolo);
WHILE simbolo <> '.' DO
    BEGIN
        IF simbolo = 'e' THEN conto := succ(conto);
        read (simbolo)
    END;
write ('Ci sono',conto :1,);
WriteLn ('"e" nella frase.')
```

La differenza tra le due strutture dovrebbe apparire piuttosto evidente. L'azione necessaria per fissare la condizione che termina il ciclo (la read nell'esempio) deve in genere essere ripetuta due volte se si usa la struttura WHILE: una prima di entrare nel ciclo e l'altra all'interno dello stesso.
Sebbene la costruzione WHILE sia l'unica veramente essenziale in qualsiasi linguaggio, talvolta è conveniente disporre di un sistema per ripetere una serie di istruzioni un numero di volte prefissato, ad esempio per una serie di valori compresi in una gamma ben precisa. Chi è abituato al BASIC si sentirà già più a suo agio su questo terreno: si tratta ovviamente del ciclo FOR...NEXT. Tuttavia occorre un po' d'attenzione, poiché in PASCAL tale struttura è leggermente diversa da quella in BASIC. In primo luogo, come contatore può essere usata qualsiasi variabile scalare (non solo quelle intere). In secondo luogo, fatto forse molto più importante, i cicli FOR...DO del PASCAL sono più affidabili: come accade nel ciclo WHILE, il ciclo può non essere eseguito affatto (se, ad esempio, il valore di partenza è superiore a quello finale) ed esistono, inoltre, alcune ferree limitazioni sull'uso delle variabili di controllo. In particolare, non è assolutamente consentito modificare il valore del contatore all'interno del ciclo. I vantaggi di quella che può apparentemente sembrare una limitazione appaiono evidenti in seguito, quando si trattano funzioni e procedure. Ecco un chiaro esempio d'illegalità in PASCAL:

```
FOR N := 1 TO 10 DO
    IF N = 10 THEN
        N := 1
```

In questo particolare esempio il ciclo risultante è senza fine, poiché N non raggiungerebbe mai il valore finale 10.
Si noti la sintassi del costrutto FOR...DO: tra le parole riservate FOR e TO c'è l'assegnazione che fissa il valore iniziale del contatore, mentre il valore finale è dato dall'espressione racchiusa tra TO e DO. Naturalmente, i due valori (iniziale e finale) devono appartenere allo stesso tipo di scalare. Ecco alcuni esempi:

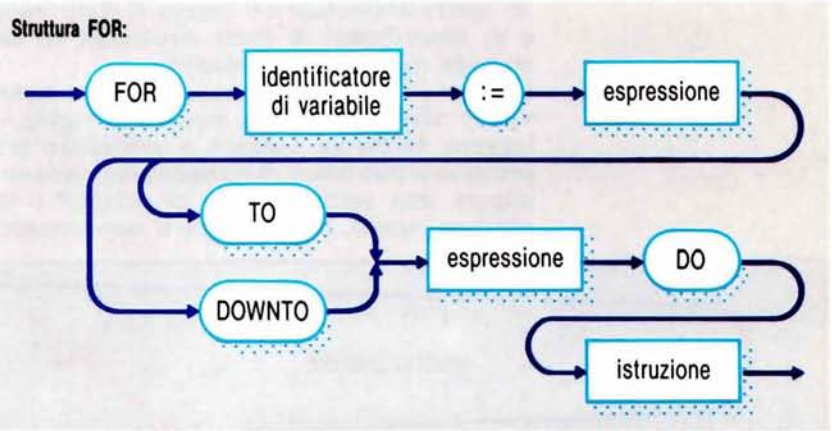
```
• FOR lettera := 'A' TO 'Z' DO {ecc.}
• FOR mese := Gen TO Dic DO {ecc.}
• FOR N := N TO succ (MaxInt DIV 1000) DO {ecc.}
```

Nell'ultimo esempio si suppone che a N sia stato in assegnato precedenza un adeguato valore (si ricordi però che se tale valore è superiore a quello finale, il ciclo non viene eseguito neppure una volta). Per eseguire cicli in ordine decrescente, al posto di TO si adopera la parola riservata DOWNTO:

```
FOR ContoAllaRovescia := 10 DOWNTO 0 DO
    WriteLn (ContoAllaRovescia :32 — 3 * ContoAllaRovescia);
WriteLn ('Fase di DECOLLO!')
```

Il PASCAL mantiene un rigido controllo sui cicli FOR. Non è prevista una clausola STEP (comune in BASIC) per fissare il passo degli incrementi o dei decrementi.

Struttura FOR:



CAFFE PASCAL...

```
PROGRAM          Caffe      (output);

TYPE
    caffe = (Bahia, Portorico, CostaDoro, Arabia);
VAR
      prezzo,
      peso      : real;
      genere    : caffe;
BEGIN
    WriteLn ('Kg' :7,'Bahia' :13, 'Portorico' :13, 'Costa d''oro' :13,
            'Arabia' :13);
    WriteLn;
    peso := 0.5; {comincia da mezzo chilo}

    REPEAT
        IF trunc (peso)>0.4
            THEN {scrive come #.#}
                Write (peso : 6 : 1, ' ')
            ELSE {scrive come intero}
                Write (round (peso) : 4, ' ' : 3);

        FOR genere := Bahia TO Arabia DO
            BEGIN
                CASE genere OF
                  Bahia     : prezzo := 6200;
                  Portorico : prezzo := 6300;
                  CostaDoro: prezzo := 6650;
                  Arabia    : prezzo := 6800
                END;      {case}
                  {stampa il prezzo}
                Write (prezzo * peso : 13 : 0);
            END;
                  {comincia una nuova linea}
        WriteLn;
        peso := peso + 0.5

    UNTIL peso>10
END.
```

Un semplice esempio

Il programma qui a fianco produce un listino di prezzi, riferiti ai quattro tipi di caffè specificati, suddiviso in scaglioni di mezzo chilogrammo alla volta. Ciascun tipo di caffè è rappresentato da una identificatore costante "concettuale", elencato nella definizione TYPE di caffe (si noti la mancanza dell'accento il cui uso, in PASCAL, è consentito soltanto all'interno di stringhe racchiuse tra apici). La selezione del prezzo abbinato a ciascun tipo di caffè è fatta, naturalmente, ricorrendo alla struttura CASE. La visualizzazione delle quantità di peso è formattata in modo da sopprimere la parte decimale non desiderata, qualora non si tratti di un valore intero. Questo esempio ha forse poca utilità pratica, tuttavia contiene molti dei concetti di programmazione fin qui esposta

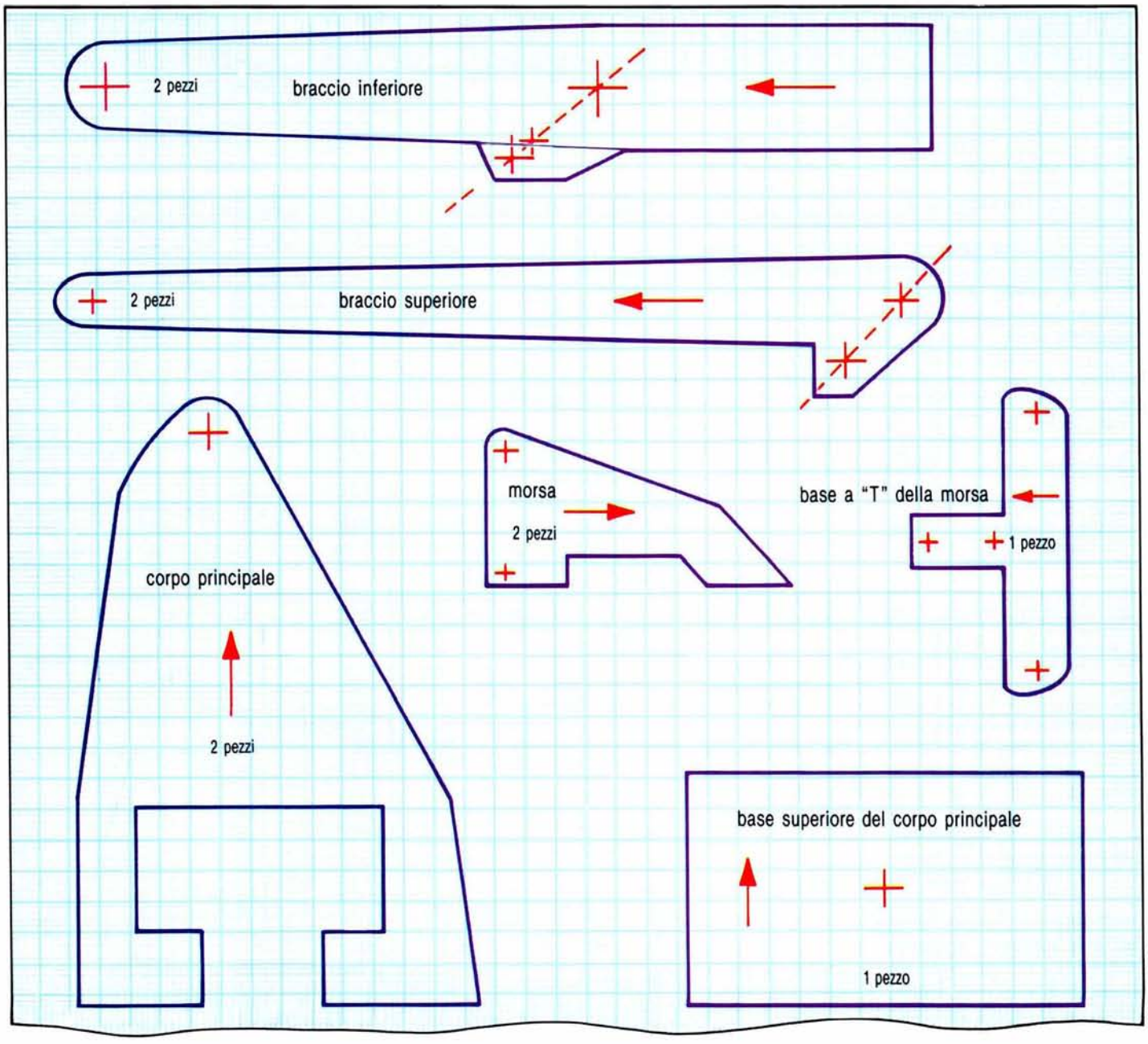
Braccio di ferro

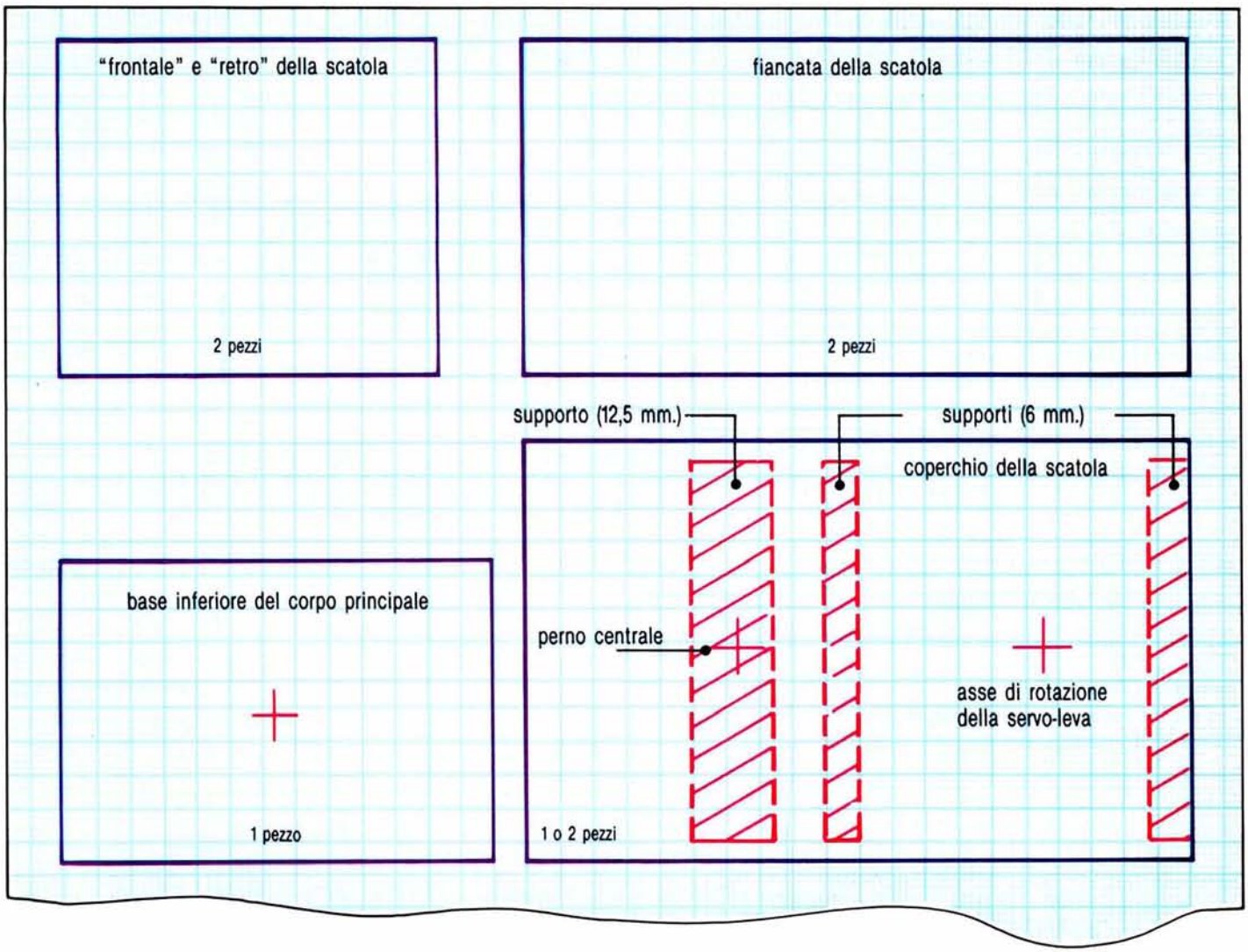
Inizia la costruzione del nostro braccio robotizzato

Le parti occorrenti per la costruzione del braccio robot hanno una varia provenienza: i componenti elettronici e le parti mobili possono essere acquistate nei negozi di elettronica e di modellismo, le parti strutturali in un comune negozio di ferramenta.
La base e i diversi segmenti del braccio robot vanno realizzati in un materiale rigido e leggero, facile da tagliare e modellare (ad esempio si può usare del comune compensato oppure una sottile lamina di perspex o di plastica rigida). Se si sceglie il compensato, occorre acquistarne un foglio di 50x50 cm., spesso 4 mm. Dopo aver riprodotto su di esso, in scala, i modelli delle diverse parti, si passa a tagliarli con un seghetto da traforo. I segmenti del braccio si ricavano da tre tavolette di legno lunghe 55 cm. (due devono essere spesse 6 mm, la terza 12,5 mm.)

Motori e parti elettroniche

Per i movimenti del braccio robot occorrono





quattro servomotori a controllo digitale, da 5v: i più adatti sono probabilmente i motori Futaba S128. Sui loro alberini devono essere fissati dei dischi di plastica del diametro di 30 mm. reperibili, assieme ai motori, in un negozio di modellismo. Volendo realizzare un braccio capace di sollevare oggetti più pesanti di un comune rocchetto di filo, è necessario adoperare motori più potenti.

I motori vengono alimentati da una corrente continua a 5v. Sebbene tutti e tre i microcomputer previsti per il controllo del robot possiedano proprie linee d'alimentazione a 5v, sulla porta d'espansione, tali tensioni sono destinate a circuiti elettronici esterni composti da unità logiche a transistor (TTL). La massima corrente disponibile si aggira quindi sui 100 mA: troppo poco per un servomotore, che a massimo carico può richiedere una corrente di öltre 200 mA. Non è consigliabile, dunque, alimentare i servomotori del braccio attraverso la porta utente. La soluzione ideale è un alimentatore a corrente continua da 5v o, come alternativa più economica, un portabatterie dotato di morsetti, in cui alloggiare tre pile da 1,5. La tensione in uscita, a 4,5v, è sufficiente per alimentare il sistema.

Le prese per i motori devono essere montate su una piastrina tipo veroboard da 9 fori x 20 piste. Il braccio robot viene collegato al computer da un cavo a nastro a 20 linee lungo circa 2 m. Inoltre, i possessori di Micro BBC o di Commodore 64 devono acquistare, rispettivamente, un raccordo IDC a 20 canali o un connettore a pettine a 24 canali, per collegare il braccio alla porta utente. Il collegamento con lo Spectrum avviene attraverso l'interfaccia per il controllo del robot, presentata a pag. 1472. Le modifiche necessarie per questa interfaccia vengono descritte in una fase successiva. Per fabbricare i perni di articolazione del braccio sono necessari sei tubi lunghi 75 mm., tre con diametro esterno di 4 mm. e tre con diametro esterno di 5 mm. Ciascun perno è composto da un tubo da 4 mm. inserito in quello da 5 mm.; due spaziatori posti alle estremità del perno tengono

MATERIALE NECESSARIO

- Viti e dadi per assemblare la morsa
- Spaziatori con diametro interno di 5 mm., 6 mm.
- Legno e colle epossidiche
- 16 viti autofilettanti lunghe 10 mm. e guarnizioni di gomma per fissare i motori
- 5 cm. di gommapiuma per le ganasce della morsa
- Stagno per saldature e nastro isolante

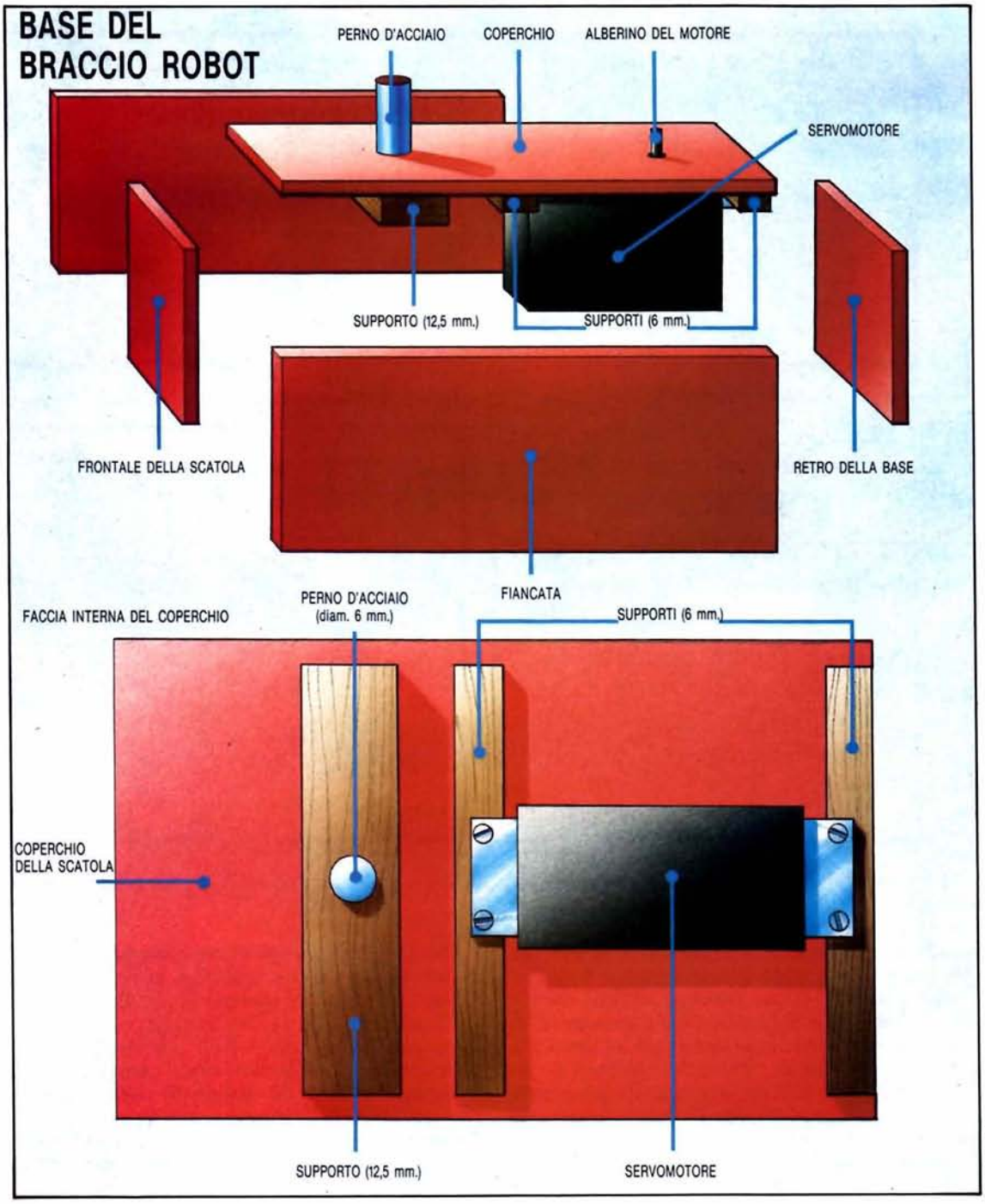
Schema dei pezzi

Dopo aver riprodotto le sagome dei pezzi sul materiale scelto per la realizzazione, si passi a tagliarli nella quantità indicata (di alcuni pezzi servono due esemplari). Smussare con cura ogni pezzo, per eliminare eventuali irregolarità o sporgenze

ASSEMBLAGGIO DELLA BASE

Incollare il supporto di 12,5 mm. sul coperchio della scatola. Praticare il foro che alloggia il perno d'acciaio. Dopo aver forato il coperchio e il supporto, inserire il perno e incollarlo. Incollare sul coperchio gli altri due supporti, sui quali viene fissato uno dei servomotori. Praticare il foro per l'alberino del servomotore. Inserire le guarnizioni di gomma e avvitare il servomotore ai supporti usando 4 viti autofilettanti. Incollare le 4 pareti della scatola. Quando la colla è asciutta, passare un foglio di carta vetrata sugli angoli e sui bordi della base





separati i due elementi consentendo loro di ruotare l'uno all'interno dell'altro.
Il corpo principale del braccio viene innestato sulla scatola di base per mezzo di un perno d'acciaio, lungo 25 mm. e con diametro di 6 mm., collegato al corpo del braccio tramite un cuscinetto a sfera. Questo è dotato di un bordo esterno ed ha un diametro interno di 6 mm. Cuscinetti simili vengono spesso usati per le ruote anteriori dei modellini di automobili e possono essere acquistati nei negozi di modellismo. Il perno, fissato saldamente al coperchio della scatola, deve ruotare liberamente all'interno del cuscinetto.
La morsa del robot è azionata da un motore posto nella parte inferiore del braccio. Il motore viene collegato alla morsa da un cavetto d'acciaio flessibile, detto talvolta anche "serpente", lungo 500 cm. Si tratta di un cavetto d'acciaio che scorre all'interno di una guaina di plastica, spesso usato sugli aeromodelli per manovrare i flap. L'apertura e la chiusura della morsa sono controllate da un collegamento rigido, lungo 50 mm., fissato all'estremità superiore del "serpente".
I segmenti del braccio vengono mossi da un sistema composto da 4 leve d'acciaio, del diametro di 2 mm. e lunghe 15 cm. Le leve possono essere collegate ai dischi dei motori direttamente o tramite piccoli giunti a snodo.

Organizzazione dei dati

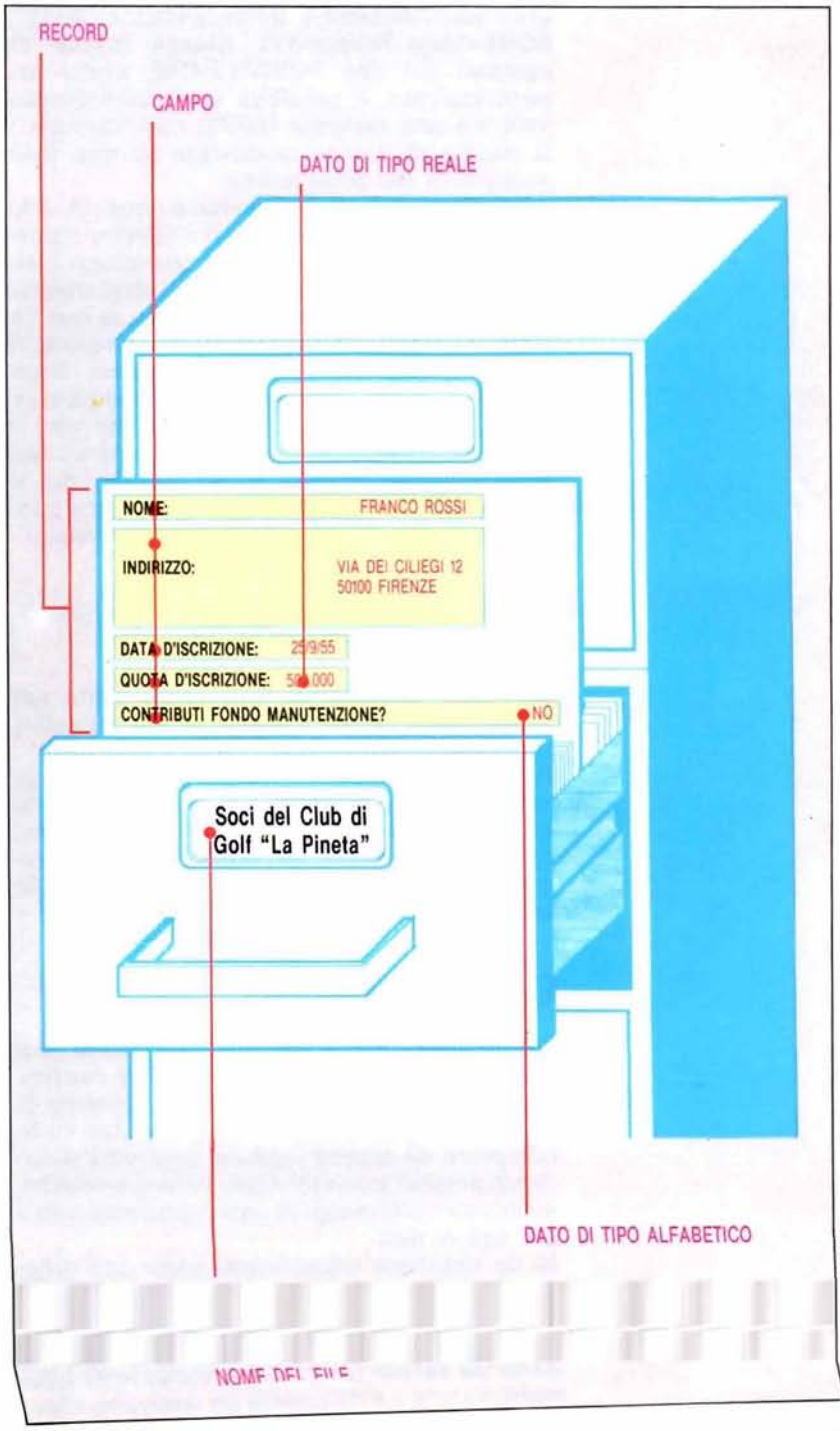
Cosa si nasconde dietro il progetto di un buon sistema di archiviazione

Segretari di club, agenti immobiliari, amministratori... tutte queste persone hanno una cosa in comune: devono trattare ogni giorno grandi quantità di dati strutturati. Un computer e un programma per la gestione di un database possono agevolare l'organizzazione e l'aggiornamento delle informazioni ma, prima di descrivere il funzionamento dei programmi di database, conviene ricordare brevemente alcuni concetti fondamentali in merito alla organizzazione dei dati.

Un programmatore di computer è solito suddividere i dati in due categorie ben distinte: quelli strutturati e quelli non strutturati. I dati non strutturati sono unità indipendenti, di solito immagazzinate in variabili o costanti. Ecco due esempi: NUMERO_DI_PROVE:=NUMERO_DI_PROVE+1 oppure IF CENTRO=1 THEN PROCEDURA INCREMENTO. In questi esempi, NUMERO_DI_PROVE e CENTRO sono variabili. Nel primo caso, il contenuto della variabile viene modificato, nel secondo, esso viene sottoposto a verifica (con eventuale salto a una procedura). Naturalmente le due variabili possono contenere un solo valore alla volta.

I programmatori, soprattutto quelli abituati a lavorare in BASIC, sono soliti considerare i dati come entità non strutturate mentre, nella realtà, ci serviamo comunemente di dati strutturati. Si pensi ad esempio ai nostri amici: le cose che sappiamo su di loro sono decisamente informazioni strutturate: ne conosciamo il nome, il sesso, l'età approssimativa, il lavoro che svolgono, forse lo stipendio e altre cose ancora. In campo informatico, una simile serie di dati tra loro collegati (riferiti a una persona, a un oggetto o a un concetto) viene detta "record". Ogni record è composto da uno o più "campi" contenenti dati dello stesso tipo o diversi. Una serie omogenea di record è chiamata "file".

Per offrirere un esempio concreto, si consideri il segretario di un club di golf con 500 soci. Per amministrare il club, egli usa uno schedario composto da schede "personalizzate". Ciascuna scheda contiene informazioni (dati) di vario genere (tipo), ad esempio: cognome, nome, sesso, età, anno d'iscrizione, quota d'iscrizione pagata?, reddito, numero di tessera, ecc. Dal punto di vista informatico, ciascun dato interno al record viene considerato un campo e i vari campi contengono tipi di dati differenti. COGNOME e NOME sono ovviamente stringhe di lettere alfabetiche quindi non possono accogliere informazioni di tipo numerico. SESSO è un campo di tipo Booleano, capace di ricevere solo uno di due valori binari. ETÀ è un campo numerico di tipo intero. Su ANNO D'ISCRIZIONE deve esistere un tipo di controllo, affinché sia accettata una ragionevole e limitata, serie di valori interi (è improbabile che l'anno di iscrizione sia, ad esempio, anteriore al 1900 o posteriore al 2100!). Infine REDDITO (qui incluso ai soli fini dell'esempio) è un campo numerico di tipo reale, capace di contenere anche cifre decimali.



I programmatori devono conoscere i tipi di dati previsti dai diversi linguaggi di programmazione. Alcuni linguaggi, come il BASIC e il C, ne possiedono un numero limitato mentre altri, come il PASCAL, richiedono che tutte le informazioni facciano riferimento a tipi di dati rigidamente definiti.
Per meglio comprendere l'importanza di questa organizzazione, vediamo come sono gestiti i dati in BASIC e in PASCAL, prima di passare alla descrizione dei database. Il BASIC riconosce soltanto due strutture per i dati: le variabili e i file. Se all'interno di un programma BASIC è necessaria un'ulteriore strutturazione, è il programmatore che deve crearla, perché il linguaggio ne è sprovvisto. Il BASIC prevede vari tipi di variabili, tra i quali i valori interi (ad esempio, PUNTEGGIO% = 7), valori reali a singola precisione (SALDO! = 12345630.25) e a doppia precisione (DISTANZA# = 12325630.66666666) e infine stringhe (PROFESSIONE$ = 'Libero Professionista'). Manca invece di costanti del tipo PIGRECO = 3.141592, anche se, naturalmente, è possibile assegnare questo valore a una variabile PIGRECO, che corre però il rischio di essere modificata in una fase successiva del programma.
Altri linguaggi di programmazione (il PASCAL ne è un ottimo esempio) offrono numerosi tipi di dati predefiniti e consentono inoltre al programmatore di crearne altri, diversi da quelli esistenti, benché basati su di essi. In PASCAL, tali tipi sono le costanti (invariabili) e le variabili del tipo char (caratteri), integer (numeri interi), real (numeri reali come 12,71) e boolean (dati binari che possono essere veri o falsi). Gli insiemi, le matrici, i record e i file sono strutture interne al linguaggio, ma il programmatore può facilmente definire nuovi tipi di dati, come nel seguente esempio:

```
TYPE
   GIORNO = (LUNEDI, MARTEDI, MERCOLEDI, GIOVEDI,
             VENERDI, SABATO, DOMENICA)
```

Qualsiasi variabile di tipo GIORNO usata nel programma può assumere solo uno dei valori specificati nella dichiarazione e non altri.
Per il programmatore, questo tipo di strutturazione presenta vantaggi e svantaggi: in alcuni casi complica le cose, ma in altri consente di evitare grossolani errori di programmazione. Passiamo ora ad esaminare le strutture di dati relative ai database.

Cos'è un database?

Un database non è che una raccolta di dati organizzati in modo strutturato. Per convenzione, i dati vengono conservati all'interno di un file (con un proprio nome), a sua volta composto da record (ognuno contraddistinto da un proprio numero). Ogni record è suddiviso in uno o più campi di dati contenenti uno o più tipi di dati.
In un database organizzato come uno schedario da tavolo la chiave principale di ogni registrazione è solitamente un cognome, classificato in ordine alfabetico. Se occorre consultare la scheda del socio Rossi, è necessario scorrere lo schedario fino al file R, individuare i nomi che iniziano per Ro e infine localizzare la scheda del signor Rossi. Fin qui niente di speciale, ma se invece fosse richiesto di contare quanti sono i soci del club di sesso femminile oppure quanti soci, al di sopra di 36 anni, hanno un reddito superiore a 12 milioni, il lavoro richiederebbe tempi incredibilmente lunghi.
Un programma per la gestione di database permette di ottenere i risultati di simili ricerche incrociate semplicemente premendo qualche tasto in più. Occorre avere ben chiara la distinzione fra database (la raccolta di dati e informazioni) e il programma che li gestisce, generalmente chiamato "database manager" o più brevemente DBM.

I Database Manager

Le prestazioni dei DBM in commercio variano notevolmente da pacchetto a pacchetto. Quelli più semplici sono poco più che piccole rubriche computerizzate, capaci di individuare un record in base a un semplice parametro del tipo NOME = ?. I DBM più complessi usano invece dei veri e propri linguaggi interni, che permettono di estrarre e gestire le informazioni contenute nel database in modo estremamente sofisticato. Spesso consentono anche il trasferimento dei dati in file adatti ad essere impiegati con altri programmi (contabilità, word processor, pacchetti statistici o grafici, ecc.).
Nelle varie parti di questa serie viene offerta una panoramica sui DBM, dai più economici ai più sofisticati, con una sommaria descrizione del funzionamento.
Tuttavia, prima di procedere oltre, torniamo per un momento al concetto di tipo di dati: un buon DBM deve consentire all'utente di definire il tipo di dato relativo a ciascun campo, rifiutare registrazioni del tipo NOME = 143326 o ANNO D'ISCRIZIONE = 1066 e visualizzare un messaggio d'errore in risposta a richieste di ricerca formulate in modo errato. Un buon DBM, diversamente dai linguaggi di programmazione, deve possedere una procedura di verifica del tipo di dato, per prevenire l'introduzione di informazioni sbagliate durante la creazione o la modifica dei record.
Le prossime parti di questa serie sono dedicate all'organizzazione dei record dei diversi DBM, ai database multi-file e relazionali, al confronto tra i campi e i record a lunghezza fissa e quelli a lunghezza variabile, alla creazione dei database e all'estrazione delle informazioni. I tre DBM esaminati con maggior attenzione sono l'*Archive* della Psion, il *dBASEII* della Ashton Tate e il *Card Box* della Caxton Software, con sintetici accenni anche ad altri pacchetti. Gli esempi riportati usano la sintassi dei software relativi, ma vengono forniti esempi generali scritti in BASCAL, uno pseudo-linguaggio derivato dal BASIC e dal PASCAL.

Gli eventi principali

Benché simili, gli "eventi" sono diversi dalle interrupt: esaminiamo le differenze

Esistono varie occasioni in cui vengono generate segnalazioni di "eventi" nel Micro BBC. Ad esempio quando un carattere viene depositato nel buffer della tastiera, dopo l'emissione del segnale di sincronizzazione verticale, una volta completata una conversione da parte del convertitore A/D o, infine, quando viene premuto il tasto Escape. Alcune di queste circostanze dipendono dall'hardware e provocano anche l'emissione di interrupt, gestite tramite il vettore IRQV1 (vedere a pag. 1558). Gli eventi, dunque, permettono di usare alcune di queste richieste di interrupt ad alta priorità accedendovi, per così dire, dalla porta di servizio. Affinché il computer possa generare eventi, questi devono prima essere opportunamente abilitati. Fatto ciò, alla prima emissione di un tale segnale la CPU salta alla routine il cui indirizzo è contenuto nel vettore situato in &220 e &221. Per evitare l'esecuzione di tale routine, è necessario disabilitare il particolare evento. L'abilitazione e la disabilitazione degli eventi avviene tramite due chiamate OSBYTE. La seguente tabella mostra gli eventi disponibili più importanti e le relative chiamate OSBYTE:

Abilitazione e disabilitazione di alcuni utili eventi

Evento	Abilitaz.	Disabil.
Il buffer di uscita si svuota	*FX14,0	*FX13,0
Il buffer d'ingresso si riempie	*FX14,1	*FX13,1
Un carattere entra nel buffer d'ingresso	*FX14,2	*FX13,2
Conversione A/D completata	*FX14,3	*FX13,3
Inizio sincronizzazione verticale	*FX14,4	*FX13,4
Intervallo scaduto nel timer interno	*FX14,5	*FX13,5
Pressione tasto Escape	*FX14,6	*FX13,6

Pertanto, per abilitare l'evento Intervallo scaduto nel timer interno, si impartisce il comando *FX14,5 in BASIC (o l'equivalente in codice macchina). Fatto ciò, la routine il cui indirizzo si trova nel vettore in &220 e &221 viene eseguita ogni volta che il timer, via via incrementato, raggiunge lo zero. Ovviamente, è indispensabile che all'indirizzo puntato dal vettore esista una routine, altrimenti il computer si "inchioda", come si dice in gergo.
Si sarà notato che esistono più eventi, ma un solo vettore per la routine di servizio: è evidente che la routine deve essere scritta in modo da poter distinguere e tener conto di ciascun evento abilitato (salvo il caso in cui l'evento abilitato sia uno solo e quindi noto a priori). Per sapere quale evento ha provocato la chiamata, va esaminato il contenuto del registro A, immediatamente all'ingresso della routine. La seguente tabella illustra il contenuto dei registri A, X e Y al loro ingresso nella routine di servizio:

Contenuto dei registri all'ingresso della routine per gli eventi

Evento	Registro A	Registro X	Registro Y
Buffer d'uscita	0	Numero buffer	#
Buffer d'ingresso	1	Numero buffer	#
Carattere nel buffer	2	#	Carattere
Convertitore A/D	3	#	Numero canale
Sincronizzazione verticale	4	#	#
Intervallo del timer	5	#	#
Pressione tasto Escape	6	#	#

In questa tabella, il segno # indica che il contenuto del registro è indeterminato.
Nel redigere una routine per la gestione di eventi conviene sempre, per prima cosa, conservare almeno il contenuto del registro A, per poterlo poi esaminare all'occorrenza. Infatti, diversamente da quanto accade nella gestione delle interrupt, dove il Micro BBC salva automaticamente i contenuti dei registri sullo stack, qui occorre che sia il programmatore a provvedere a ciò. Il ritorno dalla routine di servizio viene assicurato con una normale istruzione RTS. Fin qui la teoria: adesso passiamo ad esempi pratici su ciascun evento particolare.

- *Evento 0:* È generato quando il buffer di uscita si svuota. Il numero di buffer, qui, è lo stesso di quello usato nella chiamata OSBYTE 21 (vedere a pag. 1437).
- *Evento 1:* È generato quando il buffer d'ingresso corrente si riempie e si tenta d'immettervi ancora un altro carattere. Anche qui il numero di buffer è lo stesso di quello nella chiamata OSBYTE 21. Il valore in Y è il codice ASCII del carattere che ha causato l'evento.
- *Evento 2:* È generato ogni volta che un carattere entra nel buffer d'ingresso (in genere a seguito della pressione di un tasto). Il seguente programma mostra gli elementi basilari di una routine di servizio, generando un comando VDU 7 ogni volta che viene premuto un tasto, anche quando il programma in BASIC è in funzione.

```
10 DIM mc%(100)
20 FOR I%=0 TO 2 STEP 2
```

```
30 P% = mc%
40 [OPTI%
50 PHP
60 PHA
70 TXA:PHA
80 TYA:PHA
90
100 LDA #7
110 JSR &FFE3
120
130 PLA:TAY
140 PLA:TAX
150 PLA
160 PLP
170 RTS
180
190 ]:NEXTI%
200 ?&220 = mc% MOD 256:REM parte bassa dell'indirizzo posta
    nel vettore
210 ?&221 = mc% DIV 256:REM parte alta dell'indirizzo posta nel
    vettore
220 *FX14,2
230 REM Abilitare l'evento solo dopo aver
240 REM preparato la routine
250 REPEAT
260 PRINTI%
270 I% = I% + 1
280 UNTIL FALSE
```

- *Evento 3:* Viene generato ogni volta che uno qualsiasi dei quattro canali di conversione A/D del Micro BBC ha completato una conversione. Ciò può accadere ogni 5 o 10 millisecondi, a seconda del valore selezionato per il convertitore.
- *Evento 4:* Viene generato simultaneamente all'inizio del segnale di sincronizzazione verticale, 50 volte al secondo. Può essere utile per ottenere impulsi di temporizzazione.
- *Evento 5:* È probabilmente il più utile tra gli eventi generati dal Micro BBC, il cui sistema operativo aggiorna periodicamente non solo la variabile di sistema TIME, ma anche (ogni centesimo di secondo) uno speciale timer per intervalli. Di questo timer si è già parlato trattando delle chiamate OSWORD (vedere a pag. 1458). Poiché, quando il timer raggiunge il valore 0 il computer genera un evento, quest'ultimo è utile per interrompere con regolarità l'esecuzione di un programma ai fini di eseguire particolari compiti. Per la scrittura di valori nel timer si usa la chiamata OSWORD con A=4, per la lettura si adopera OSWORD con A=3.

Se vogliamo che il computer generi un evento dopo un secondo, occorre depositare nel timer un valore pari al massimo valore consentito, diminuito di 100: poiché il timer viene via via incrementato, esso raggiungerà lo zero dopo 100 "scatti". Il valore massimo per il timer, che è composto da 5 byte, è &FFFFFFFFFF. Ecco un programma che dimostra l'impiego di questo timer per generare un "bip" a intervalli regolari:

```
10 DIM mc%(100), ritardo% 10
20 FORI% = 0 TO 2 STEP 2
30 P% = mc%
40 [OPTI%
50 PHP:PHA
60 TXA:PHA
70 TYA:PHA
80 LDA #7:JSR &FFE3
90 .regola orologio /aggiusta il valore per la successiva interru-
   zione
100 LDX #ritardo% MOD 256 /prepara il blocco per la OSWORD
110 LDY #ritardo% DIV 256
120 LDA #4
130 JSR &FFF1
140 PLA:TAY
150 PLA:TAX
160 PLA:PLP
170 RTS
180 ]:NEXTI%
190 ?&220 = mc% MOD 256
200 ?&221 = mc% DIV 225
210 !ritardo% = &FFFFFF9C:ritardo%?4 = &FF
220 *FX14,5
230 CALL regola orologio:REM avvia il timer
240 END
```

- *Evento 6:* Viene generato premendo il tasto di Escape.

Gli altri tre eventi disponibili sono: per segnalare un errore sul sistema della porta RS423, un errore sul sistema Econet e, infine, un evento definito dall'utente.

APPLICHIAMO LE NOSTRE CONOSCENZE

Il seguente programma illustra alcuni dei principi esposti fin qui nella serie dedicata al SO del Micro BBC. Si esamini attentamente il listato, poi lo si trascriva e si dia un RUN. Si tenti di scoprire (1) quale operazione viene svolta dal programma e (2) come funziona il programma. Le relative spiegazioni sono presentate nella prossima parte di questa serie. (Suggerimento: il nome della variabile alla linea 80 dovrebbe mettere sulla buona strada.)

```
 20 PROCassembla
 30 END
 50 DEFPROCassembla
 70 DIM MC% &FF
 80 vecchiovett=?&20E+256*?&20F
 90 FOR pass=0 TO 3 STEP 3
100 P%=MC%
120 [OPT pass
130 .temp EQUB 00
140 .inizio
150 PHP
160 STA temp
170 TXA:PHA
180 TYA:PHA
200 JSR verif
230 PLA:TAY
240 PLA:TAX
250 LDA temp
260 PLP
270 JMP vecchiovett
290 .verif
300 LDA temp
310 CMP#65
320 BMI out
330 CMP #91
340 BPL out
350 ADC #32
360 STA temp
370 .out RTS
390 ]:NEXT
400 ?&20E=inizio MOD 256
410 ?&20F=inizio DIV 256
420 ENDPROC
```

Una banca dati in classe

Creare e gestire un archivio su computer è un'attività didattica da non sottovalutare

Avere a disposizione una "biblioteca elettronica" porta numerosi vantaggi, tra i quali un accesso immediato alle informazioni e la comodità di non doversi spostare da un luogo all'altro (ad esempio dalla classe alla biblioteca). Per di più le informazioni conservate nella memoria del computer possono essere aggiornate costantemente, mentre quelle riportate da un testo tradizionale sono "statiche" e possono essere modificate soltanto se il libro viene ristampato. Altro fattore da considerare è il costo dei testi scolastici, immancabile oggetto di contestazioni ad ogni apertura di un nuovo anno scolastico. La carta stampata, nonostante quanto appena affermato, ha pur sempre un ruolo insostituibile e non va certo eliminata, ma va invece affiancata e integrata con un coscienzioso impiego dei mezzi che l'attuale tecnologia offre.

Altri vantaggi divengono maggiormente evidenti nell'uso di computer nella telematica, per accedere a banche dati di notevole capacità. Avere, come si dice in gergo, "in linea" un'intera enciclopedia risulta estremamente utile. Nel caso di una ricerca condotta con metodi tradizionali l'unica chiave di accesso alle informazioni è rappresentata dall'indice dell'opera (magari da quello "ragionato"). Fin qui nulla da eccepire: basta prelevare l'apposito volume dallo scaffale e annotarsi i vari riferimenti. Ma nel caso di un'enciclopedia conservata su supporti magnetici e accessibile con un computer le potenzialità si estendono in modo pressoché illimitato. La ricerca, infatti, non è più circoscritta a quanto gli

Il mondo intorno a noi

Molti credono che l'impiego di computer nell'ambito scolastico costringa gli allievi a lunghe ore passate di fronte allo schermo dell'apparecchio. Al contrario, le più recenti esperienze evidenziano come (specialmente nell'uso di programmi per la creazione e la gestione di archivi) le attività coinvolte sono in massima parte svolte lontano dal computer. La necessità di raccogliere informazioni da inserire nell'archivio, infatti, può addirittura prevedere camminate all'aperto, alla ricerca di fiori o animali da catalogare, stimolando così anche lo spirito d'osservazione



STEVE CROSS

autori hanno ritenuto opportuno inserire e catalogare nell'indice: si può chiedere al software del computer di produrre un elenco di riferimenti relativi a un qualsiasi argomento, magari applicando criteri selettivi (ad esempio la stampa di un elenco di scrittori appartenenti ad una determinata corrente letteraria, vissuti in un particolare periodo storico o in una limitata zona geografica). Ovviamente, anche se ciò è teoricamente possibile, non conviene memorizzare l'intero contenuto dello scibile umano su normali dischetti da home o personal computer: nonostante tutta la buona volontà il numero dei dischetti sarebbe talmente alto da precludere una qualsiasi praticità di accesso. Ecco dove entra in gioco la telematica: basta un normale home computer e un economico modem per accedere alle colossali banche dati universitarie o commerciali.

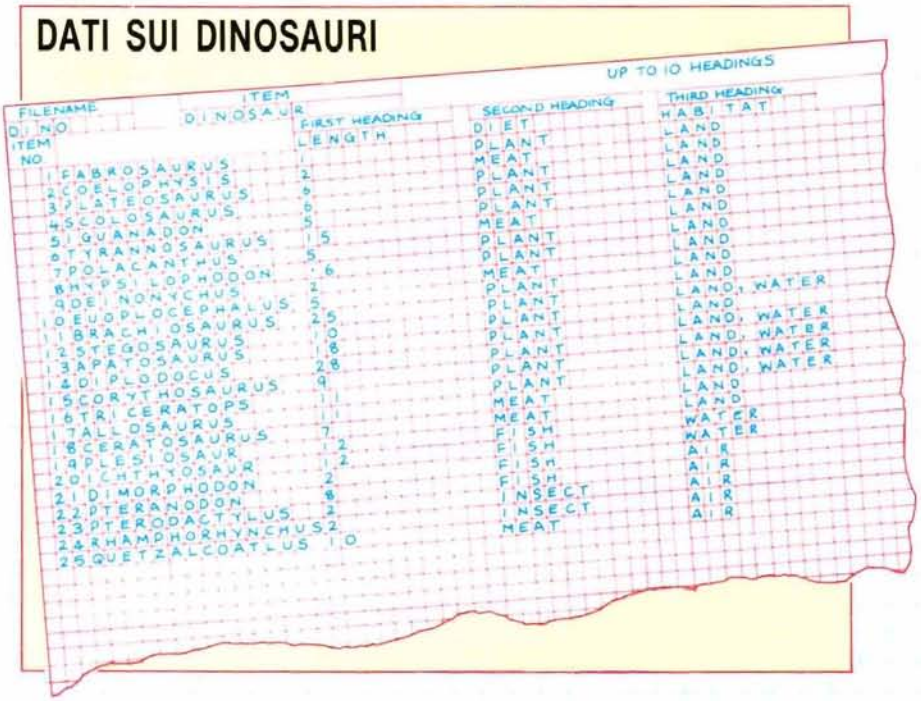
Il numero e la qualità di questi database accessibili al pubblico va crescendo di giorno in giorno (basti pensare ai servizi di Videotel e Televideo operanti da tempo anche in Italia).

Negli Stati Uniti esiste uno degli archivi computerizzati più vasti del mondo, il Dialog, costantemente consultato da innumerevoli organizzazioni e istituti scolastici. Esso contiene oltre 200 argomenti di base, che si sviluppano su un arco di ben oltre 100 milioni di elementi d'informazione: ognuno di questi corrisponde a un riferimento o ad un estratto sui più disparati argomenti, dall'arte alla zoologia. Infatti, durante la consultazione del Dialog, non vengono fornite soltanto indicazioni bibliografiche (nome dell'autore, titolo dell'opera, ecc.) ma anche brevi brani riassuntivi sul contenuto della pubblicazione, più che sufficiente per formarsi una precisa idea sull'opportunità o meno di approfondire l'argomento con la lettura della pubblicazione. Naturalmente, quanto compare sul video del computer può essere riprodotto sulla propria stampante o, nel peggiore dei casi, può essere richiesta per posta una copia su carta delle informazioni occorrenti.

Un altro servizio che vale la pena di citare è quello offerto alle scuole britanniche, all'interno della rete Prestel, chiamato *Schools Link*. Si tratta, in effetti, di una sorta di "palestra culturale computerizzata", in cui vengono raccolte e scambiate informazioni di ogni genere, purché inerenti al settore scolastico o didattico in genere. Per esser più precisi, questo servizio è destinato particolarmente agli insegnanti, che possono trovarvi aggiornate recensioni sia sull'hardware sia sul software prodotti dall'industria informatica e utilizzabili nell'ambito scolastico.

Tutto ciò fa presagire un radicale e quanto mai prossimo cambiamento dei processi educativi e forse dello stesso ruolo della scuola, anche se i tempi occorrenti affinché questa tecnologia venga accettata e assorbita efficacemente sono certamente lunghi.

Nel frattempo, però, esiste già tutta una serie di programmi di più modeste ambizioni, ma non per questo di minore efficacia didattica.



Una memoria... magnetica

Il programma "Factfile" offre agli studenti un completo sistema per la creazione e la gestione di un archivio, su qualsiasi argomento a scelta. Il package comprende anche un esempio applicativo "Dino", che contiene informazioni sui dinosauri. Il manuale consiglia di impratichirsi con questo archivio dimostrativo, prima di passare alla creazione di un database completamente nuovo. Il programma originale, come si vede dalle illustrazioni, è in lingua inglese, ma il principio è facilmente applicabile anche in lingue diverse, come l'italiano

Opzione di ricerca



Il menu di "Dino"

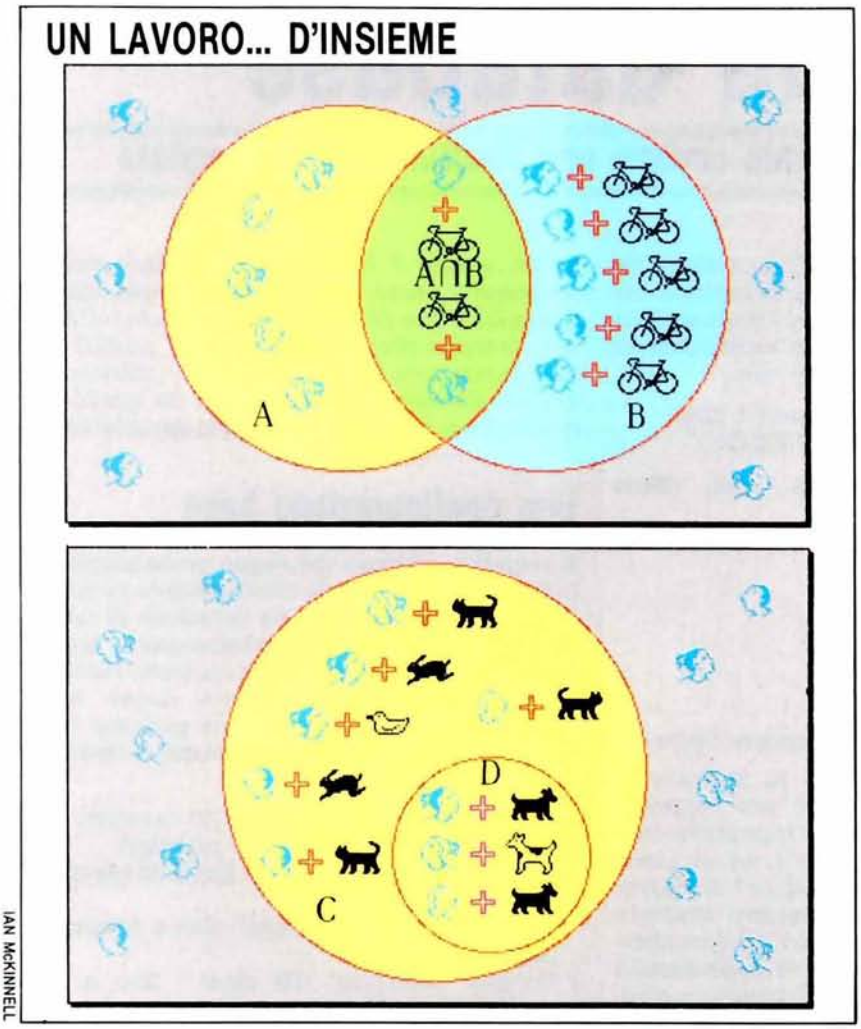


In particolare va diffondendosi sempre più l'uso di piccoli archivi computerizzati, da creare e da gestire "ad hoc" limitatamente ad una classe di allievi. Questi programmi hanno il pregio di offrire notevoli spunti per quanto concerne la catalogazione, la correlazione e i criteri di accesso alle informazioni, abituando al tempo stesso gli studenti all'uso dei computer.

Il programma "Factfile"

Come già detto più volte, la nazione europea che più si è dedicata, fin da principio, all'applicazione dell'informatica nel settore scolastico è la Gran Bretagna. Un progetto che ha ottenuto un notevole successo nelle scuole d'oltremanica è quello sviluppato da un congresso tenutosi all'Università di Cambridge nel 1981: il *Factfile*. Questo consiste, essenzialmente, in un programma che consente anche agli allievi delle scuole inferiori di creare e di gestire un database su qualsiasi argomento a loro scelta.

Dalle numerose esperienza fatte, si è potuto notare che l'efficacia di un simile sistema non tarda a stimolare l'interesse e l'apprendimento degli studenti. Supponiamo che una classe di alunni intenda creare un archivio contenente informazioni sugli insegnanti. Per prima cosa occorre stabilire le categorie nelle quali suddividere le informazioni e ciò dipende dalle conoscenze già acquisite e dalla fantasia dei singoli individui. Ad esempio possono essere previste categorie quali: nome, cognome, età, altezza e "tipo di insegnante". Fissata la struttura generale del database gli studenti devono poi occuparsi di raccogliere i dati da inserire nell'archivio e ciò implica attività di vario genere. Ad esempio, volendo includere dati fisici quali l'altez-



za, va stabilita l'unità di misura da usare e poi procedere alla misurazione pratica. Il processo diventa ancor più complesso via via che gli allievi approfondiscono il significato della suddivisione in categorie che loro stessi hanno generato. Il campo "tipo di insegnante", ad esempio, può offrire lo spunto per animate discussioni: va riferito alla materia insegnata oppure a un commento sulla minore o maggiore "bontà" della persona?

Creata la struttura di base dell'archivio e immessi i dati, inizia un altro genere di esperienza: l'accesso ragionato e la correlazione delle informazioni memorizzate. Ad esempio il database può essere interrogato per fornire un elenco degli insegnanti maschi di una particolare materia al di sotto di una certa età. L'analisi dei risultati ottenuti dalle varie richieste va a tutto vantaggio dello sviluppo del processo apprenditivo. *Factfile*, ovviamente, consente di conservare su dischetto o su cassetta tutto il lavoro fatto, permettendo così di consultare e aggiornare l'archivio in qualsiasi momento successivo.

Una critica spesso rivolta a questo genere di esperienze è che i bambini trascorrono troppo tempo davanti al video del computer, lontani quindi dal contatto col "mondo esterno": ciò è vero solo in parte, poiché l'uso di programmi quali il *Factfile* implica, come già accennato, lo svolgimento di una miriade di altre attività collegate, gran parte delle quali svolte senza il diretto uso del computer. Se l'archivio creato concerne informazioni sulla flora locale, ad esempio, i bambini trascorreranno parte del tempo a raccogliere campioni, disegni o foto di piante, per poi studiare il criterio in base al quale catalogare i vari dati.

Un altro programma per database molto usato è il *Your Facts*, che funziona in modo ancor più elementare. Sullo schermo compaiono semplici domande relative a ciascun alunno (il nome, il sesso, se possiede un cane, un orologio, una bicicletta, ecc.). Terminata l'immissione dei dati per un allievo, il programma chiede se ve ne sono altri, fino ad arrivare alla massima capienza dell'archivio, 40 bambini, sufficiente quindi anche per classi numerose. Conclusa la fase di immissione, viene presentato il menu principale con opzioni per visionare i record, per eseguire ricerche selettive su particolari categorie (chi possiede una bicicletta?) e per avviare un gioco. Questo gioco consiste di una serie di domande (del tipo Possiedi un orologio?, Sei un maschio o una femmina?) che il computer pone al bambino nel tentativo di indovinare il suo nome. Questo package è un'ottima introduzione all'uso dei database ed incoraggia l'apprendimento della lettura e della scrittura.

Informazioni classificate

La creazione e la gestione di un archivio non sono totalmente avulse dalle altre attività scolastiche. Ad esempio i concetti dell'insiemistica, che stanno alla base di una qualsiasi catalogazione di dati, vengono oggi introdotti fin dalla più tenera età. La possibilità di eseguire ricerche selettive, all'interno di un database, non fa che mettere in pratica alcuni di questi concetti sugli insiemi, in particolare quello relativo all'intersezione di due insiemi (figura in alto) e quello concernente i sottoinsiemi (figura in basso) in cui un insieme è totalmente incorporato in un altro

Il programma "One World"

Una software house australiana, la Active Learning Systems, ha realizzato un database piuttosto ampio, il *One World*, che contiene informazioni geografiche e sociali (distribuite in oltre 30 categorie) su tutte le nazioni del mondo. Ciò che rende attraente questo programma, come del resto tutti i database, è la possibilità di correlare i dati memorizzati. Il menu principale ha questo aspetto:

1. VISIONE DATI DI UNA NAZIONE
2. RICERCA SU UNA CATEGORIA
3. ANALISI SU CRITERI DIVERSI
4. RIFERIMENTI VARI
5. FINE

Se un allievo si ricorda il nome di Yitzhak Shamir, quale leader di un governo, ma non il paese di appartenenza, può selezionare l'opzione 2 e immettere il nome sotto la categoria CAPI DI GOVERNO. Il programma risponde comunicando che YITZHAK SHAMIR è il capo di governo di ISRAELE.

L'opzione ANALISI è la più versatile e consente ricerche selettive su un massimo di 20 categorie di dati. Si pensi, ad esempio, che se un allievo desidera conoscere la percentuale di popolazione rurale in una particolare nazione le categorie coinvolte sono solo due: la nazione e la percentuale della popolazione. *One World* è un eccellente strumento per insegnare ad accedere e ad interpretare nella corretta prospettiva sia dati generici che dati statistici.

Usare un database

Per un archivio efficiente occorre una struttura ben congegnata

Uno dei vantaggi delle vecchie rubriche da scrivania è la loro semplicità d'impiego, soprattutto per quanto riguarda l'ordinamento alfabetico degli indirizzi. Nulla vieta infatti di inserire una registrazione del tipo:

EUGENIO FIORI, Corso Magenta 256,
Milano CAP. 20100 Tel. 02-732456

— Carla, la sua ragazza - Tel. ufficio 456372
— chiamare dopo le 4 - cambia casa in Gennaio

seguita da:

FEDERICO - 433545

e da:

FRANCO - vedi Concessionaria Sperry

Su una rubrica, si è liberi di indicare le diverse persone per nome o per cognome oppure di usare una sorta di "indirizzamento indiretto" con "puntatori" riferiti ad altri indirizzi. Nel trasferire un "database" di questo genere su computer, però, occorre adottare un sistema di registrazione più rigido e sistematico. Poiché una rubrica computerizzata presenta numerosi svantaggi rispetto a quella su carta, è necessario progettare delle maschere di registrazione versatili, capaci di far fronte alle più svariate esigenze. Esaminiamo i diversi campi separatamente, iniziando da quello del "Nome".
Questo termine generico viene comunemente usato per indicare cognome e nome proprio di una persona. Nella maggior parte dei paesi europei di solito il nome precede il cognome ma, in alcune nazioni (ad esempio in Cina, in Giappone, in Ungheria e, un tempo, anche in Italia) accade il contrario. Di ciò occorre tener conto per non creare confusione. È ovvio, pertanto, che va fissato una volta per tutte un criterio di registrazione: inserire prima il cognome oppure il nome. Inoltre è necessario assegnare al campo una dimensione adeguata, in grado di accogliere anche i nominativi più lunghi. Se il programma di gestione del database (per brevità DBM, da DataBase Manager) usa campi di lunghezza fissa, occorre scegliere quello le cui dimensioni siano adeguate (i campi a lunghezza fissa presentano altri svantaggi, che vengono esaminati in seguito). Dovendo definire l'articolazione interna della rubrica, come di qualsiasi altro database, probabilmente conviene usare il campo del cognome come campo chiave principale e iniziare ogni registrazione partendo dal cognome, seguito da uno o più nomi propri. Anche il campo dell'indirizzo richiede una particolare attenzione, poiché il formato dell'indirizzo varia da paese a paese. In Italia, ad esempio, viene usato il Codice di Avviamento Postale o CAP, non sempre previsto dai sistemi postali di altri Paesi, mentre in Giappone l'indirizzo è seguito dal numero di distretto, da quello di quartiere e, infine, dal nome del destinatario.

Una configurazione base

Il semplice esempio che segue probabilmente non si adatta a tutte le combinazioni possibili: rappresenta quindi una soluzione di compromesso (qualsiasi schematizzazione implica un qualche tipo di compromesso). Partendo dal presupposto di non dover mai registrare il nominativo di una persona con 16 nomi, vediamo quale può essere la struttura di base di record:

- Campo "Cognome" - fino a 40 caratteri
- Campo "Nome" - fino a 80 caratteri
- Campo "Indirizzo" (I riga) - fino a 60 caratteri
- Campo "Indirizzo" (II riga) - fino a 80 caratteri
- Campo "Indirizzo" (III riga) - fino a 80 caratteri
- Campo "Indirizzo" (IV riga) - fino a 80 caratteri
- Campo "Indirizzo" (V riga) - fino a 80 caratteri
- Campo "Telefono" - fino a 20 caratteri
- Campo "Note" - fino a 80 caratteri

Questo schema (in gergo "maschera") per rubrica computerizzata dovrebbe adattarsi alla maggior parte degli indirizzi, ma non è detto che sia del tutto privo di inconvenienti. Uno di questi, almeno potenzialmente, è la ridotta dimensione del campo "Note". Un secondo problema può derivare dalla mancanza di un campo specificamente dedicato alla nazione. A seconda della lunghezza dell'indirizzo, infatti, il nome della nazione può finire nella terza, nella quarta, nella quinta riga, oppure mancare del tutto. D'altro canto la collocazione di questa indicazione provoca problemi soltanto nel caso che il campo "Nazione" serva come chiave di ricerca del database. Se la rubrica è destinata a contenere molti indirizzi stranieri, conviene creare una maschera con un campo "Nazione" separato. Tali decisioni devono essere prese fin dall'inizio, in fase di progettazione.
Le particolari caratteristiche di funzionamento di un database variano, ovviamente, in base al tipo di DBM usato. Uno dei DBM più semplici è il *Card Box* della Caxton che, pur possedendo modeste funzioni per l'estrazione

e la gestione delle informazioni, risponde in modo adeguato alle normali esigenze della gestione di indirizzari. Il *Card Box* non offre sofisticati linguaggi di programmazione per la manipolazione dei campi e dei record, tuttavia prevede una serie di semplici comandi, inseribili da tastiera, per l'estrazione di record specifici. Impiegando il *Card Box* per creare il database descritto in precedenza, è sufficiente caricare il programma ed eseguire una semplice sequenza di operazioni.

Uso del "Card Box"

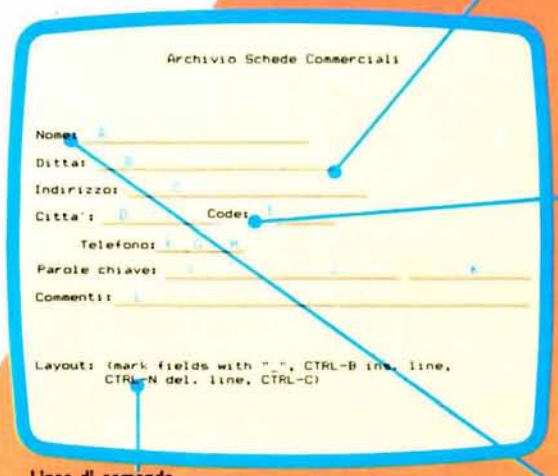
Per prima cosa occorre definire il formato del record, che influisce sul "tipo" di informazioni registrabili, sul tipo di "indicizzazione" (quale campo designare come campo "chiave") e sul formato delle maschere di schermo o di stampa. Supponiamo di assegnare all'archivio il nome RUBRICA: il *Card Box* crea automaticamente un "file di formato", chiamato RUBRICA.FMT, destinato a contenere le informazioni di base sul formato, appunto, del database. Tale file può essere editato per modificare la maschera di visualizzazione delle informazioni e, addirittura, è anche possibile creare file di formato alternativi, per visualizzare le informazioni del database in modi diversi.
Come la maggior parte dei DBM, anche il *Card Box* consente di abbinare scritte permanenti (chiamate "etichette") a ciascun campo. Nel caso di una rubrica, il significato dei campi è quasi sempre evidente: "Eugenio Fiori" è chiaramente un nome e non una parte dell'indirizzo, mentre 02-732456 è quasi certamente il numero telefonico. Se le informazioni sullo schermo non sono troppo fitte, conviene introdurre delle scritte permanenti, che chiariscano il significato di ciascuno di essi. Confrontiamo i seguenti record:

06166
3995
86
12,25
Vite a testa piatta

MATRICOLA	06166
CODICE DI LISTINO	3995
GIACENZE DI MAGAZZINO	86
PREZZO	12,25
DESCRIZIONE	Vite a testa piatta

Le informazioni contenute nei due record sono identiche, ma le etichette presenti nei campi del secondo, riducono le probabilità di commettere errori.
Il *Card Box* permette di assegnare a ciascun campo uno tra quattro attributi di indicizzazione: NONE (nessuno), MAN (manuale), AUTO (automatico) e ALL (tutti). In un database per un magazzino, difficilmente il PREZZO verrà usato come campo chiave, è più probabile che questo ruolo venga svolto dal campo MATRICOLA oppure CODICE DI LISTINO.
Tornando all'esempio della rubrica computerizzata, è logico supporre che come chiave venga adoperato il cognome e, tutt'al più, anche il nome, per cui i relativi campi devono essere trasformati in campi "chiave". Se però la rubrica viene adoperata per usi commerciali, può essere utile una ricerca dei record che adopri come chiavi il nome della città o il prefisso telefonico. È importante ricordare che i criteri di accesso ai database computerizzati sono diversi da quelli adottati per i vecchi schedari manuali, ma soprattutto che essi offrono possibilità pressoché illimitate. Ecco perché la creazione di un database veramente efficiente, richiede idee molto chiare sul suo futuro impiego.

Fase di progettazione
Il "Megafinder" consente un'ampia libertà nella creazione delle "maschere". La maschera dell'immagine superiore deriva da quella fornita come standard dal programma, mostrata nella figura inferiore



Linea di comando
Alcuni dei comandi consentiti durante la progettazione: in questa fase è possibile inserire e cancellare i caratteri. Al termine, la maschera può essere salvata su disco premendo CTRL-C



Lunghezza di campo
Viene indicata dai caratteri di sottolineatura (_). La lunghezza totale di tutti i campi di una maschera determina il numero di maschere (ossia record) memorizzabili in un file. Conviene quindi creare campi di lunghezza limitata, tenendo conto del tipo di dati che essi devono accogliere

Etichette di campo
Queste vengono inserite dall'utente nel corso creazione della maschera e, in questo programma, servono per puro riferimento. Il programma usa le etichette di campo durante la visualizzazione e l'immissione di informazioni sullo schermo ma le ignora durante il riordinamento dei dati, poiché per questa operazione ricorre agli identificatori di campo

Identificatori di campo
Da non confondersi con le ETICHETTE DI CAMPO, gli identificatori determinano l'ordine d'inserimento delle informazioni nei diversi campi. Inoltre consentono di riordinare i campi singolarmente o in gruppi. Usando la funzione INDEX del "Megafinder" per assegnare il campo D all'Indice 1, l'utente può riordinare rapidamente i record, ad esempio, in base alla classificazione alfabetica delle città

Le presento la sua scheda
I sistemi di gestione dei database offrono svariati metodi di manipolazione delle informazioni. Questa immagine tratta dal "Megafinder" mostra un record dell'archivio schede commerciali e, presso il margine inferiore, un elenco delle opzioni di comando. Alle informazioni contenute nell'archivio possono essere abbinati quattro diversi indici (comando INDEX) per accelerare le operazioni di richiamo. Il DBM permette inoltre di confrontare (MATCH) i dati in ingresso e quelli già archiviati, di saltare (JUMP) ad una specifica sezione del database (ad esempio, quella che contiene i nomi delle società che cominciano per M), di modificare (CHANGE) e cancellare (DELETE) le informazioni e di visualizzarle (PRINT) su schermo o su stampante

Pronti a salpare

Sta per iniziare il viaggio verso il Nuovo Mondo: abbiamo viveri a sufficienza?



Durante il viaggio, le razioni settimanali vengono controllate all'interno di un ciclo FOR...NEXT, in cui la variabile WK funge da contatore delle settimane di viaggio. Il limite superiore del ciclo è indicato dal valore contenuto nella variabile JL, posta a 8 all'inizio del programma. Internamente al ciclo vengono chiamate le subroutine che gestiscono le diverse operazioni da ripetersi settimanalmente. La prima riguarda il giornale di bordo del capitano, sul quale vengono registrati il "rendimento" dell'equipaggio, il livello delle scorte, ecc. Questo ciclo è compreso tra le linee 800 e 889.

Il cibo è il fattore principale da cui dipende il rendimento di ciascun membro dell'equipaggio. Per mantenersi in perfetta forma, un marinaio deve consumare l'intera razione settimanale di frutta, ortaggi, carne e acqua. All'inizio del viaggio, al rendimento di ciascun membro dell'equipaggio viene assegnato il valore 100, depositato nella matrice TS(,). Il programma verifica settimanalmente se le scorte delle diverse provviste sono sufficienti per l'intero viaggio. Quando uno dei cibi comincia a scarseggiare, il programma chiede al giocatore se vuole dimezzarne le razioni. Questo taglio delle razioni può essere mantenuto solo per una settimana e riduce il rendimento della ciurma di 5 unità.

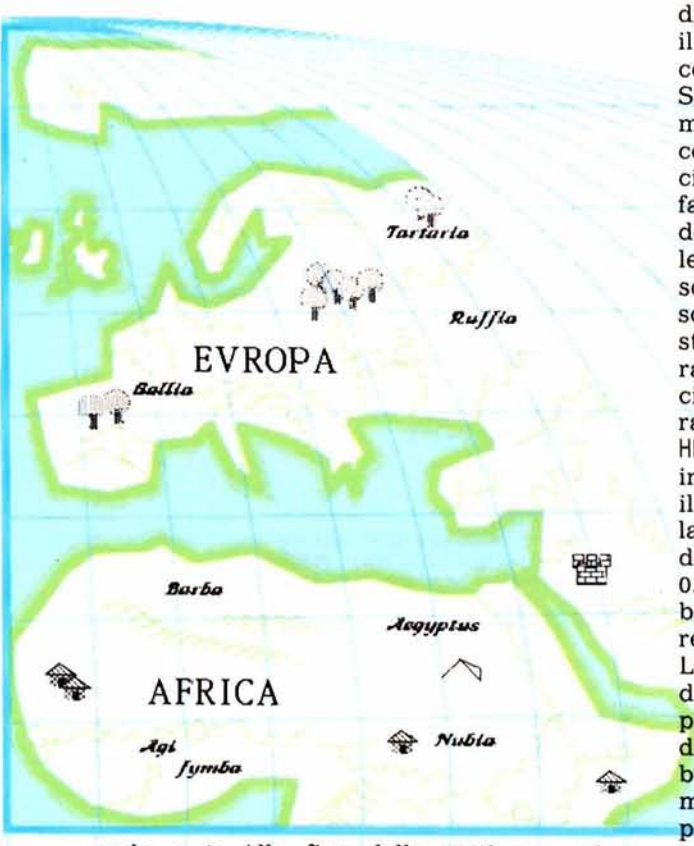
Quando una delle provviste si esaurisce, il rendimento di ogni marinaio decresce di 10 unità alla settimana. Se, ad esempio, terminano le scorte di frutta e viene dimezzata la razione d'acqua, si ha un calo di rendimento di 15 unità alla settimana. Nel caso in cui anche l'acqua si esaurisca, il calo è di 20 unità. Se, dopo un razionamento di una settimana, il livello delle scorte di cibo e d'acqua consente di tornare alle razioni intere fino alla fine del viaggio, queste vengono ripristinate automaticamente all'inizio della settimana successiva.

Lo stato di salute dell'equipaggio viene registrato sul giornale di bordo all'inizio di ogni settimana: di ciò si occupa la subroutine alla linea 4000. Lo stato di salute è classificato in 4 livelli, assegnati in base al rendimento. Il più alto è "in perfetta salute" (rendimento 75-100), seguito da "in salute" (50-75), "malato" (25-50) e "molto malato" (1-25). Quando il rendimento raggiunge lo 0, il marinaio muore e il suo corpo viene calato in mare; tuttavia la sua paga settimanale continua a essere accantonata (va consegnata ai familiari quando la nave farà ritorno in porto). Questa subroutine calcola anche le paghe per la settimana successiva, l'ammontare totale delle paghe dall'inizio del viaggio e il denaro rimanente. Il totale delle paghe accumulate viene azzerato all'inizio della subroutine (linea 800): ad ogni passaggio nel ciclo, ossia ad ogni settimana, questo totale viene aggiornato con il totale settimanale delle paghe. Qualora la cifra da versare in paghe superi il denaro ancora in cassa (il confronto viene eseguito dalla subroutine alla linea 5000), il giocatore viene informato che i profitti del viaggio verranno (almeno in parte) usati per pagare l'equipaggio.

Se tutto va bene, il viaggio termina dopo 8 settimane ma alcune situazioni impreviste possono accrescerne la durata. La linea 801 crea una variabile stringa, H$, che indica l'eventuale riduzione delle razioni. Il ciclo iterativo alla linea 4060 esamina la matrice TS(,) per determinare la categoria e il rendimento di ciascun membro della ciurma. Dato che la matrice può contenere informazioni per un massimo di 16 membri dell'equipaggio, il ciclo termina quando T raggiunge il valore 16. Se un particolare elemento della "categoria" vale 0, il corrispondente elemento del "rendimento" della stessa matrice è vuoto, quindi il ciclo passa all'elemento successivo. In caso contrario, la linea 4075 visualizza la categoria e il rendimento del membro dell'equipaggio.

Morte di un marinaio

Quando un marinaio muore, il decesso deve essere registrato sul giornale di bordo. Ma poiché un rendimento pari a 0 significa appunto "morte", il programma potrebbe credere erroneamente che ogni elemento vuoto, all'interno della matrice, corrisponda ad un decesso. Questo inconveniente viene eliminato assegnando, per l'arco di una settimana, il valore −999 all'elemento il cui rendimento



cala a 0. Alla fine della settimana viene ripristinato il valore 0 e il programma comunica il decesso all'inizio della settimana seguente. Grazie a questo stratagemma i membri deceduti non continuano a comparire nell'elenco settimanale dell'equipaggio. Le linee 4080-4098 confrontano il rendimento dei marinai in vita con i 4 parametri relativi allo stato di salute e la linea 4099 visualizza le condizioni di ciascuno. Il ciclo termina alla linea 4110, che rimanda il programma al successivo elemento della matrice.

Le paghe settimanali vengono calcolate da un secondo ciclo (linee 4120-4135), ripetuto per ciascuna categoria dell'equipaggio. Le paghe per le diverse categorie vengono calcolate alla linea 4130, moltiplicando il numero di categoria per il livello di paga e registrando il risultato nella variabile WW. Creata e azzerata all'inizio di ogni settimana, WW rappresenta la paga settimanale e viene visualizzata dalla linea 4145 per essere poi sommata al totale corrente delle paghe, WT, visualizzato dalle linee 4160 e 4165. Il giocatore può così confrontare questo valore con il denaro in cassa.

La subroutine alla linea 5100 gestisce le razioni per l'equipaggio; usa un ciclo FOR...NEXT, ripetuto per ciascun tipo di razione, ed esegue una serie di controlli sulla quantità di razioni disponibili. Per prima cosa verifica che il livello di ciascuna provvista, registrato in PA() sia maggiore di 0. In caso contrario, riduce di 10 unità il rendimento di tutti i marinai. Questa operazione viene eseguita dalla subroutine alla linea 9300, che sottrae al rendimento la quantità WF. L'uso una variabile, anziché di un valore fisso, permette di utilizzare la subroutine per decrementare il rendimento di una qualsiasi quantità necessaria: basta passarla in WF.

Se l'ammontare delle provviste nella stiva è maggiore di 0, viene eseguito un ulteriore controllo per stabilire se queste sono sufficienti fino al termine del viaggio. Ciò viene fatto moltiplicando il numero dei membri dell'equipaggio (CN) per il consumo settimanale di provviste e per il numero delle rimanenti settimane di viaggio. Se la quantità delle scorte residue è minore del fabbisogno previsto, il giocatore può decidere di dimezzare le razioni di una o più provviste. Il livello di ciascuna scorta (sia a razioni intere che a razioni dimezzate) è indicato dalla matrice HR(), dimensionata alla linea 802. A razioni intere, gli elementi della matrice contengono il valore 1; quando la razione di una particolare provvista viene dimezzata, il corrispondente elemento della matrice viene posto a 0.5. Se le razioni vengono dimezzate, la subroutine alla linea 9300 riduce di 5 unità il rendimento di tutti i membri dell'equipaggio.

La verifica finale che precede la consegna delle razioni controlla se il livello di una particolare provvista è inferiore alla quantità da distribuire. In questo caso, dopo la distribuzione, al corrispondente elemento della matrice PA() viene assegnato il valore −999 per indicare che la provvista è esaurita.

Basic & Basic

Spectrum:

Eseguire le seguenti modifiche:

```
 847 LET I$=INKEY$:IF I$=""THEN GO TO 847
4010 CLS
4190 LET I$=INKEY$:IF I$=""THEN GO TO 4190
4205 CLS
4295 LET I$=INKEY$:IF I$=""THEN GO TO 4295
4305 CLS
4398 LET I$=INKEY$:IF I$=""THEN GO TO 4398
5010 CLS
5080 LET I$=INKEY$:IF I$=""THEN GO TO 5080
5103 CLS
5220 INPUT I$=LET I$=I$(1TO1)
5298 LET I$=INKEY$:IF I$=""THEN GO TO 5298
```

Micro BBC:

Eseguire le seguenti modifiche:

```
4010 CLS
4190 I$=GET$
4205 CLS
4295 I$=GET$
4305 CLS
4398 I$=GET$
5010 CLS
5298 I$=GET$
```

MODULO 4: IL VIAGGIO

Variabile per la durata del viaggio

```
40 JL=8:REM DURATA DEL VIAGGIO
```

Ciclo principale del viaggio

```
800 WT=0
801 H$="N": REM INDICATORE RAZIONI DIMEZZATE
802 DIM HR(4):HR(1)=1:HR(2)=1:HR(3)=1:HR(4)=1
820 FOR WK=1 TO JL: REM CICLO PRINCIPALE DEL VIAGGIO
825 GOSUB 4000: REM RAPPORTO SULLA SALUTE DELL'EQUIPAGGIO
830 GOSUB 4200: REM RAPPORTO SULLE PROVVISTE
835 GOSUB 4300: REM RAPPORTO SULLE ALTRE MERCI
840 GOSUB 9200:PRINT CHR$(12)
842 PRINT :PRINT :PRINT
843 S$="PROBABILMENTE IL VIAGGIO♦": GOSUB 9100
844 PRINT "DURERA' ANCORA";JL-WK;" SETTIMANE"
845 GOSUB 9200
846 PRINT:S$=K$:GOSUB 9100
847 GET I$=INKEY$:IF I$="" THEN 847
850 GOSUB 5000: REM VERIFICA CONTEGGIO DELLA PAGA
855 GOSUB 5100: REM DISTRIBUZIONE DELLE RAZIONI
889 NEXT WK
```

Rapporto sulla salute dell'equipaggio

```
4000 REM RAPPORTO SULLA SALUTE DELL'EQUIPAGGIO
4010 PRINT CHR$(12)
4020 S$="     GIORNALE DI BORDO♦": GOSUB 9100
4025 S$="     ----------------♦": GOSUB 9100
4030 GOSUB 9200
4035 PRINT "ALL'INIZIO DELLA SETTIMANA";WK
4040 S$="LO STATO DELL'EQUIPAGGIO E'♦": GOSUB 9100
4045 GOSUB 9200:PRINT
4055 PRINT
4060 FOR T=1 TO 16
4070 IF TS(T,1)=0 THEN 4110
4075 PRINT C$(TS(T,1));"      (";
4078 IF TS(T,2)=-999 THEN S$="MORTO !!!!!!!!)♦": GOTO 4099
4080 IF TS(T,2)>75 THEN S$="IN PERFETTA SALUTE)♦": GOTO 4099
4085 IF TS(T,2)>50 THEN S$="IN SALUTE)♦": GOTO 4099
4095 IF TS(T,2)>25 THEN S$="MALATO ! )♦": GOTO 4099
4098 S$="MOLTO MALATO  !!  )♦"
4099 GOSUB 9100:GOSUB 9200
4110 NEXT T
4115 GOSUB 9200:PRINT
```

Conteggio paga settimanale

```
4119 WW=0
4120 FOR T=1 TO 5
4130 WW=WW+(CC(T)*WG(T))
4135 NEXT T
4140 S$="PAGA SETTIMANALE: ♦": GOSUB 9100
4145 PRINT WW;" MONETE D'ORO"
4150 GOSUB  9200
4155 WT=WT+WW
4160 S$="TOTALE PAGHE FINO AD ORA:♦":GOSUB 9100
4165 PRINT WT;" MONETE D'ORO"
4170 GOSUB 9200
4175 PRINT "DENARO RIMASTO = ";MO;" MONETE D'ORO"
4180 PRINT:S$=K$:GOSUB 9100
4190 GET I$=INKEY$:IF I$="" THEN 4190
4199 RETURN
```

Rapporto sulle scorte

```
4200 REM RAPPORTO SULLE PROVVISTE
4205 PRINT CHR$(12)
4206 PRINT "ALL'INIZIO DELLA SETTIMANA ";WK: GOSUB 9200
4210 S$="LA QUANTITA' DI PROVVISTE♦":GOSUB 9100
4215 S$="RIMASTE E':♦": GOSUB 9100
4220 PRINT:GOSUB 9200
4225 FOR T=1 TO 4
4226 IF PA(T)=0 OR PA(T)=-999 THEN 4240
4230 PRINT PA(T);U$(T);" DI ";P$(T)
4232 X=INT(PA(T)/(CN*PN(T)))
4235 PRINT " (SUFFICIENTI PER ";X;" SETTIMANE)"
4239 GOSUB 9200
4240 NEXT
4290 PRINT:S$=K$:GOSUB 9100
4295 GET I$=INKEY$:IF I$="" THEN 4295
4299 RETURN
```

Rapporto sulle altre merci

```
4300 REM RAPPORTO SULLE ALTRE MERCI
4305 PRINT CHR$(12)
4306 PRINT "ALL'INIZIO DELLA SETTIMANA ";WK: GOSUB 9200
4310 S$=" CI SONO ANCHE:♦": GOSUB 9100
4320 PRINT : GOSUB 9200
4322 IF OA(1)=0 THEN 4332
4325 PRINT OA(1);: S$=" BOTTIGLIE DI MEDICINALE♦":GOSUB 9100
4330 GOSUB 9200
4332 IF OA(2)=0 THEN 4342
4335 PRINT OA(2);: S$=" ARMI♦": GOSUB 9100
4340 GOSUB 9200
4342 IF OA(3)=0 THEN 4352
4345 PRINT OA(3);: S$=" SACCHI DI SALE♦": GOSUB 9100
4350 GOSUB 9200
4352 IF OA(4)=0 THEN 4362
4355 PRINT OA(4);: S$=" PEZZE DI STOFFA♦": GOSUB 9100
4360 GOSUB 9200
4362 IF OA(5)=0 THEN 4372
4365 PRINT OA(5);: S$=" COLTELLI♦": GOSUB 9100
4370 GOSUB 9200
4372 IF OA(6)=0 THEN 4380
4375 PRINT OA(6);: S$=" MONILI♦": GOSUB 9100
4380 GOSUB 9200:PRINT
4382 PRINT "SONO RIMASTE IN CASSA ";MO;: S$=" MONETE D'ORO♦":GOSUB 9100
4383 GOSUB 9200
4397 PRINT: S$=K$:GOSUB 9100
4398 GET I$=INKEY$:IF I$="" THEN 4398
4399 RETURN
```

S/R Conteggio della paga

```
5000 REM VERIFICA CONTEGGIO PAGA
5005 IF WT> MO THEN 5010
5008 GOTO 5099
5010 PRINT CHR$(12)
5020 PRINT:PRINT:PRINT
5025 S$="TRA L'EQUIPAGGIO CORRE VOCE♦": GOSUB 9100
5030 S$="CHE NON HAI DENARO SUFFICIENTE♦": GOSUB 9100
5035 S$="PER PAGARE GLI UOMINI ALLA FINE♦": GOSUB 9100
5040 S$="DEL VIAGGIO.♦": GOSUB 9100
5045 GOSUB 9200:PRINT
5050 S$="L'EQUIPAGGIO E' FURIOSO!!♦": GOSUB 9100
5055 GOSUB 9200:PRINT
5060 S$="SPERIAMO CHE TU RIESCA A FARE♦": GOSUB 9100
5065 S$="BUONI GUADAGNI !♦": GOSUB 9100
5066 GOSUB 9200
5070 PRINT:S$=K$:GOSUB 9100
5080 GET I$=INKEY$:IF I$="" THEN 5080
5099 RETURN
```

S/R Distribuzione delle razioni

```
5100 REM DISTRIBUZIONE RAZIONI
5103 PRINT CHR$(12)
5105 S$="  DISTRIBUZIONE DELLE RAZIONI♦": GOSUB 9100
5106 S$="  --------------------------♦": GOSUB 9100
5107 GOSUB 9200: PRINT "SETTIMANA:";WK: PRINT
5108 H$="N"
5110 FOR T=1 TO 4
5112 HR(T)=1
5115 IF PA(T) > 0 THEN 5180
5120 PRINT "NON C'E' PIU' ";P$(T);" !!!!!": GOSUB 9200
5130 S$="LA CIURMA SI STA INDEBOLENDO!!♦":GOSUB 9100
5135 WF=10:GOSUB 9300
5139 GOTO 5290
5180 X=(PN(T)*CN)*(JL-WK+1)
5185 IF PA(T)<X THEN 5200
5190 GOTO 5270
5200 PRINT "STANNO CALANDO LE SCORTE DI ";P$(T)
5205 GOSUB 9200
5210 S$="VUOI DIMEZZARE LE RAZIONI DI♦": GOSUB 9100
5215 PRINT P$(T);" PER L'EQUIPAGGIO ?"
5220 INPUT I$:I$=LEFT$(I$,1)
5221 IF I$<>"S" AND I$<>"N" THEN 5220: REM RISPOSTA NON CORRETTA
5225 IF I$="N" THEN 5270
5230 HR(T)=.5: H$="S"
5240 WF=5:GOSUB 9300
5250 S$="LA CIURMA SI STA INDEBOLENDO!!♦": GOSUB 9100
5270 X=PN(T)*HR(T)*CN
5272 IF X>PA(T) THEN X=PA(T)
5275 PA(T)=PA(T)-X
5280 IF PA(T)=0 THEN PA(T)=-999
5285 PRINT "DISTRIBUITI";X;U$(T);" DI ";P$(T)
5290 PRINT:GOSUB 9200:NEXT
5295 S$=K$:GOSUB 9100
5298 GET I$=INKEY$:IF I$="" THEN 5298
5299 RETURN
```

Riduzione di rendimento dell'equipaggio

```
9300 REM RIDUCE DI WF IL RENDIMENTO DELL'EQUIPAGGIO
9310 FOR S1=1 TO 16
9320 TS(S1,2)=TS(S1,2) - WF
9330 IF TS(S1,2) < 1 THEN TS(S1,2)=-999
9340 NEXT
9399 RETURN
```

Il Sanyo MBC-550

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

Sebbene la Sinclair Research abbia dimostrato, con il suo QL, che è possibile produrre un computer a 16 bit mantenendo i costi relativamente bassi, in generale i prezzi dei micro a 16 bit permangono elevati. Tale situazione sembra quasi dipendere dal tipo di utenti interessati a questi prodotti, che appaiono disposti a pagare cifre considerevoli pur di soddisfare le proprie esigenze gestionali.
Questa ipotesi trova una conferma nei computer basati sui microprocessori della serie Intel 8088 ed è probabilmente collegata alla politica commerciale della IBM, compagnia leader del settore.
La politica dei prezzi seguita dalla maggior parte dei fabbricanti ha tuttavia lasciato scoperta una vasta area di mercato che è stata occupata dai computer gestionali basati sullo Z80 e dotati di unità a dischetti che consentono l'impiego del CP/M. Solo di recente alcune compagnie hanno cercato di produrre macchine più economiche basate sull'8088.
Uno di questi apparecchi è l'MBC-550 della Sanyo, teoricamente non compatibile con il PC IBM ma, di fatto, capace di utilizzare il suo sistema di archiviazione su disco MS-DOS.
L'MBC-550, che costa circa la metà del PC-IBM, viene prodotto in due versioni: con unità a dischi singola o doppia.
Come molti altri computer gestionali, l'MBC-500 si compone di due elementi separati (la tastiera e l'unità centrale), in questo caso rivestiti da una custodia in metallo e plastica, con rifiniture color argento.
La tastiera è divisa in tre sezioni.
Sulla sinistra ci sono 5 tasti di funzione programmabili. Abbinati al tasto Shift, essi producono un massimo di 10 funzioni che dipendono dall'applicazione in uso.
In BASIC questi tasti possono essere associati a parole di comando di uso comune quali LIST, LOAD o RUN.
La tastiera alfanumerica è ben disegnata e tutti i tasti di controllo si trovano nella posizione consueta. I tasti Backspace "distruttivo" e Insert/Delete, entrambi presenti, sono attivi alternativamente, a seconda dell'applicazione.
Il tasto Return è quattro volte più grande dei tasti normali. A destra della barra spaziatrice c'è un tasto Graphics, per la visualizzazione dei caratteri grafici.
Sul lato destro della tastiera è stata inserita una tastierina numerica utilizzabile come calcolatrice oppure, abbinata al tasto Number Lock, come controllo del cursore per l'editing dei listati BASIC.
L'unità centrale, alloggiata in una custodia metallica larga e piatta, ha le dimensioni di un video registratore. Su di essa viene appoggiato il monitor (che può essere quello prodotto dalla stessa Sanyo).
Sul frontale c'è l'interruttore d'accensione e spazio per due unità a dischetti (il modello rappresentato in figura è la versione a singolo drive).
Il Sanyo MBC-550 usa floppy disk da 5 1/4 pollici. Sul davanti di ogni unità a dischetti c'è la solita levetta che blocca il dischetto e ne impedisce l'estrazione quando la testina è in attività.

Interfacce per le periferiche

Sulla parte posteriore del computer si trovano la presa d'alimentazione e le interfacce per le periferiche. A sinistra, sul pannello posteriore ci sono: il cavo di alimentazione, l'alloggiamento del fusibile e la boccola di massa. Sulla destra c'è un connettore per la

Una linea sobria

Il Sanyo MBC-550, un computer gestionale a 16 bit dal prezzo relativamente contenuto, che usa il famoso sistema operativo MS-DOS è prodotto in due versioni: l'MBC-550 con una singola unità a dischetti e l'MBC-555 con un'unità a dischetti doppia. Il monitor che si vede in figura, anch'esso prodotto dalla Sanyo, è opzionale



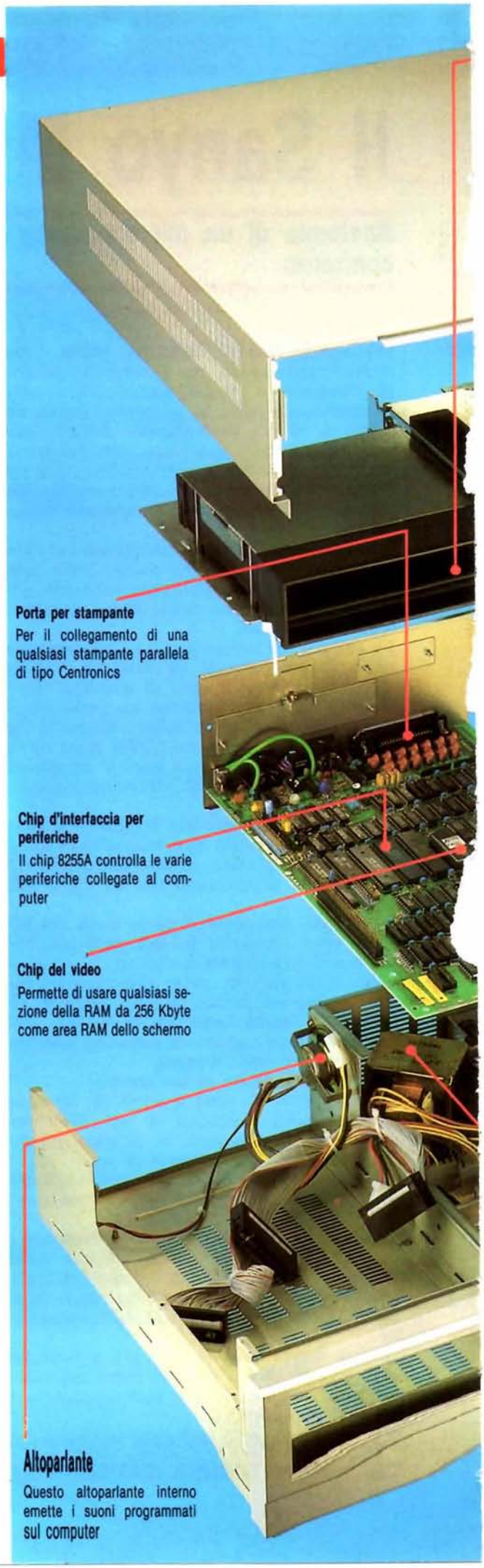
CHRIS STEVENS

porta parallela per stampante tipo Centronics e due uscite video: la prima è una presa RBG per monitor a colori, la seconda un connettore a spinotto per video composito adatto al monitor monocromatico (a fosfori verdi) prodotto dalla Sanyo.
Sopra queste interfacce sono visibili altre porte d'espansione destinate ad accogliere interfacce aggiuntive.
La porta Line, situata sopra il connettore per la stampante, può alloggiare un'interfaccia seriale RS232C che permette all'MBC-550 di comunicare con altri computer, attraverso un modem, o con altre periferiche seriali quali, ad esempio, una stampante. Alla sua destra c'è una porta per joystick della serie MBC o Apple-compatibili, che tuttavia può accogliere altre periferiche di controllo come trackball, paddle e via dicendo.
Infine c'è una finestrella che è normalmente chiusa che consente di accedere al bus di sistema e di interfacciare così periferiche d'altro tipo.
Il manuale per l'utente messo a disposizione dalla Sanyo, diversamente da quelli di molte altre compagnie, offre una serie completa di istruzioni per il montaggio di tutte le interfacce, dei banchi RAM supplementari e della seconda unità a dischetti sul modello a singolo drive.

SPAZIO PER ESPANDERSI

Sebbene l'MBC-550 sia privo del corredo d'interfacce in dotazione su alcuni dei più costosi apparecchi oggi sul mercato, possiede tuttavia una notevole espandibilità. È predisposto, tra l'altro, per l'aggiunta di interfacce per joystick e per altre periferiche seriali. Le istruzioni per il loro montaggio sono contenute nel manuale per l'utente





Porta per stampante
Per il collegamento di una qualsiasi stampante parallela di tipo Centronics

Chip d'interfaccia per periferiche
Il chip 8255A controlla le varie periferiche collegate al computer

Chip del video
Permette di usare qualsiasi sezione della RAM da 256 Kbyte come area RAM dello schermo

Altoparlante
Questo altoparlante interno emette i suoni programmati sul computer



Il resto del manuale, egualmente esauriente, contiene un glossario dei termini informatici, un breve corso didattico per il BASIC della Sanyo, comprese le istruzioni relative alla gestione BASIC del disco, e numerose informazioni di carattere tecnico.

Il BASIC della Sanyo, una versione derivata dal Microsoft, è indubbiamente soddisfacente, ma si dimostra lento nell'elaborazione delle funzioni aritmetiche. Questo limite dipende soprattutto dalla ridotta velocità di calcolo dell'8088 (a cui viene spesso associato il co-processore matematico 8087). Nonostante ciò, l'MBC sembra essere assai più lento di altri computer basati sullo stesso chip, anche per quanto riguarda il tracciamento delle linee. Infatti, poiché il BASIC Sanyo è privo del comando DRAW, per tracciare una linea occorre usare l'istruzione PSET, che "accende" un singolo pixel alla volta in una determinata colorazione.

Per visualizzare una linea è quindi necessario l'impiego di un ciclo iterativo.

Oltre ai comandi dello standard Microsoft come MID$, LLIST e CIRCLE, il BASIC Sanyo possiede una serie di istruzioni per definire e "aprire" finestre sullo schermo. Inoltre la presenza del costrutto condizionale WHILE...WEND favorisce una programmazione di tipo strutturato.

Molte parole riservate del BASIC possono essere immesse in forma abbreviata, semplicemente usando il tasto Control assieme ad altri due o tre tasti.

Ad esempio, per introdurre la parola riservata DIM basta premere contemporaneamente CTRL, SHIFT e D, mentre l'istruzione PRINT viene inserita digitando CTRL e C. Sebbene questa tecnica d'immissione presenti aspetti positivi, è difficile pensare di poter memorizzare tutte le sequenze di tasti delle 40 parole riservate del BASIC Sanyo: conviene quindi limitarsi ai comandi di uso più comune.

Il Sanyo MBC-550 è corredato da un disco di sistema contenente, oltre all'MS-DOS, il BASIC, il notissimo word processor *WordStar* e il programma di lavagna elettronica *CalcStar*.

Questi programmi applicativi sono accompagnati da un manuale che ne illustra il funzionamento.

Si potrebbe pensare che la versione base dell'MBC-550, che impiega il processore 8088 e il sistema operativo MS-DOS, incontri poche difficoltà nell'elaborare l'abbondante software prodotto per il PC-IBM. Al contrario, dopo alcune verifiche, va detto che purtroppo i limiti imposti dall'hardware della macchina sembrano impedire l'impiego di qualsiasi pacchetto IBM-compatibile.

Computer gestionali funzionanti sotto MS-DOS, capaci di eseguire software IBM-compatibile e, soprattutto economici, non deludono le aspettative di gran parte dei potenziali acquirenti, ma l'industria sembra restia a produrli.

Il Sanyo MBC-550 potrebbe rappresentare un primo tentativo in tal senso, se non fosse per la sua incompatibilità coi prodotti IBM.

SANYO MCB-550 e 555

DIMENSIONI

mm. 375 x 355 x 108

CPU

Intel 8088, funzionante a 3,6 MHz

MEMORIA

128 Kbyte di RAM, espandibile a 256 Kbyte

VIDEO

Modo testuale: 25 righe per 80 colonne
Modo grafico: 640 x 200 pixel. In alta risoluzione sono disponibili fino a 8 colori

INTERFACCE

Porta parallela tipo Centronics. Opzionalmente sono disponibili: l'interfaccia seriale RS232 e un'interfaccia per joystick di tipo Atari

LINGUAGGI FORNITI

BASIC Sanyo

TASTIERA

Tipo QWERTY, con 65 tasti. 5 tasti funzione programmabili (per un massimo di 10 funzioni) e una tastierina numerica a 19 tasti

DOCUMENTAZIONE

Il manuale è di ottima qualità e contiene una esauriente spiegazione di tutti i comandi e le istruzioni del BASIC Sanyo, anche se non contiene una vera e propria sezione didattica. Vengono date anche informazioni sul sistema operativo MS-DOS e sui programmi applicativi (inclusi) WordStar e CalcStar. Sorprendentemente, per un manuale destinato all'utente, vengono fornite dettagliate istruzioni per il montaggio delle interfacce e delle memorie aggiuntive

PRO

Il Sanyo MCB-550 è un buon apparecchio, adatto principalmente a chi desideri possedere un 16 bit senza spendere cifre troppo elevate

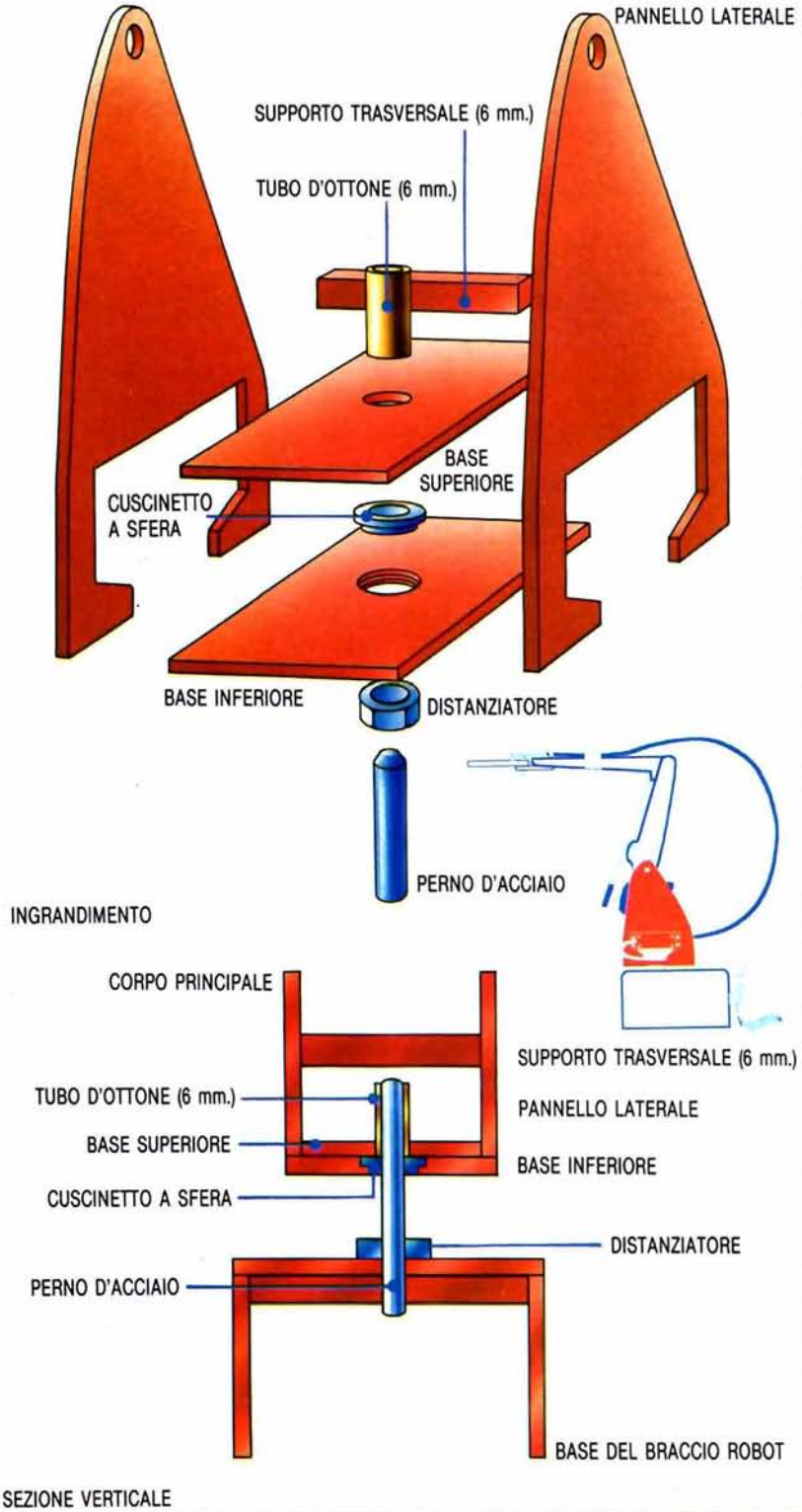
CONTRO

Considerata l'incompatibilità nei confronti del PC-IBM, nonostante la presenza del sistema operativo MS-DOS, questo apparecchio presenta un inconveniente comune a tutti i nuovi computer: manca un adeguato supporto di software

Montiamo il braccio

L'assemblaggio del corpo centrale e delle due sezioni mobili

FASE 1: MONTAGGIO DEL CORPO PRINCIPALE



Fase 1: Montaggio del corpo principale

Praticare un foro nella base inferiore (la più larga delle due) per alloggiarvi il cuscinetto a sfera. Con un trapano a punta larga o una lima, smussare il margine superiore del foro per adattarlo al bordo sporgente del cuscinetto. Praticare un secondo foro nella base superiore, per il tubo d'ottone (lungo 25 mm., diametro interno 6 mm.). Inserire il cuscinetto nella base inferiore e incollare le due basi, facendo attenzione ad allineare con precisione i due fori. Introdurre il tubo d'ottone dall'alto e fissarlo con adesivo.

Quindi montare i pannelli laterali del corpo principale. Forare gli apici dei due pannelli e inserire nei fori, a pressione, un segmento di tubo con diametro esterno di 4 mm.: il tubo deve combaciare perfettamente con entrambi i fori. Inserire tra i due pannelli un supporto trasversale di legno (sezione quadrata 6x6 mm., lungo 37 mm.), posizionandolo posteriormente e in linea con le aperture per i motori praticate nei pannelli laterali. Fissare i pannelli e il supporto con colla. Quando la colla è secca, inserire un distanziatore (ad esempio, un dado) nel perno d'acciaio sporgente dalla base (vedere a pag. 1574), quindi innestare il perno nel cuscinetto a sfera e controllare che il corpo del braccio robot possa ruotare liberamente sulla sua base.

Fase 2: Montaggio del braccio inferiore

Tagliare un segmento di tubo lungo 37 mm. (diametro esterno 7 mm.). Forare centralmente i due elementi del braccio inferiore e inserirvi a pressione il tubo. Questo non deve avere gioco nei fori poiché rappresenta la parte fissa dell'articolazione della spalla. Praticare un secondo foro sulle estremità affusolate dei due elementi e inserirvi un tubo d'ottone con diametro esterno di 4 mm. che va fissato ai fori.

Inserire adesso nei due fori centrali un secondo tubo lungo 5 mm., usando come distanziatore uno dei servomotori (vedere la figura a pag. seguente). Tagliare due supporti di legno (sezione quadrata di 6x6 mm., lunghezza 20 mm.) e disporli trasversalmente tra i due elementi del braccio inferiore. Sui due supporti, incollati rispettivamente al di sotto del tubo e presso l'estremità inferiore del braccio, viene montato il servomotore. Quando la colla è asciutta, fissare il servomotore con 4 viti autofilettanti, interponendo la guarnizione di gomma. Terminato il montaggio del braccio inferiore, fissarlo al corpo principale tramite l'articolazione del gomito. Quindi inserire un tubo lungo 45 mm. (diametro 4 mm.) attraverso i fori superiori del corpo principale e il tubo centrale del braccio inferiore.

Centrare con precisione il braccio inferiore facendolo scorrere sul tubo interno. Marcare la posizione esatta, smontare e incollare il tubo di diametro maggiore sul braccio inferiore. Quando la colla è secca, riassemblare l'articolazione della spalla.

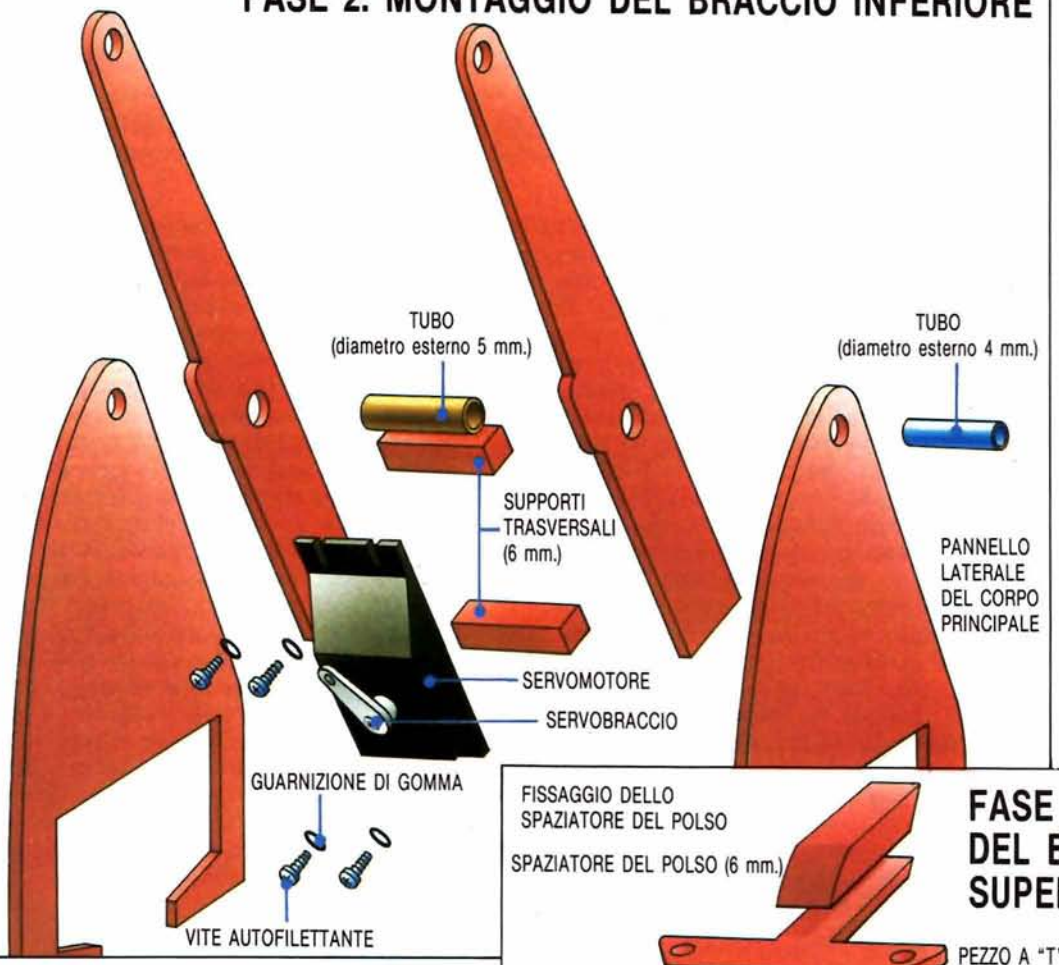
Fase 3: Montaggio del braccio superiore

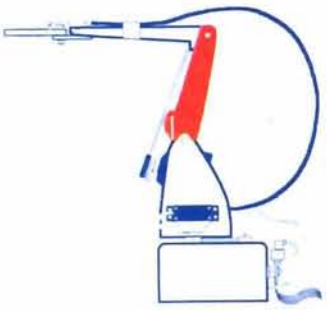
Forare le estremità più larghe degli elementi del braccio superiore e innestare un segmento di tubo lungo 5 mm. Praticare altri due fori di minor diametro sulle estremità opposte, per l'inserimento di un bulloncino. Incollare un supporto di legno (sezione quadrata di 6x6 mm., lungo 25 mm.) sul pezzo a "T" che agisce da distanziatore del polso e sagomarne le estremità. Inserire la sezione verticale del pezzo a "T" tra i due elementi del braccio superiore e forarla. Questo foro viene attraversato dal bulloncino che collega il distanziatore all'estremità superiore del braccio. Fissare il pezzo a "T" con una rondella e un dado.

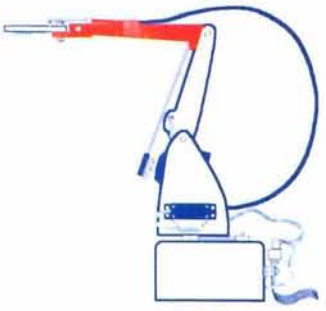
All'altra estremità inserire un tubo d'ottone lungo 17 mm. (diametro 5 mm.) nei fori praticati. Quindi collegare il braccio superiore a quello inferiore tramite l'articolazione del gomito, facendo passare un segmento di tubo lungo 25 mm. (diametro 4 mm.) attraverso l'articolazione. Dopo aver centrato il braccio superiore, smontare l'articolazione e incollare il tubo d'ottone di diametro maggiore ai due elementi del braccio superiore.

FASE 2: MONTAGGIO DEL BRACCIO INFERIORE

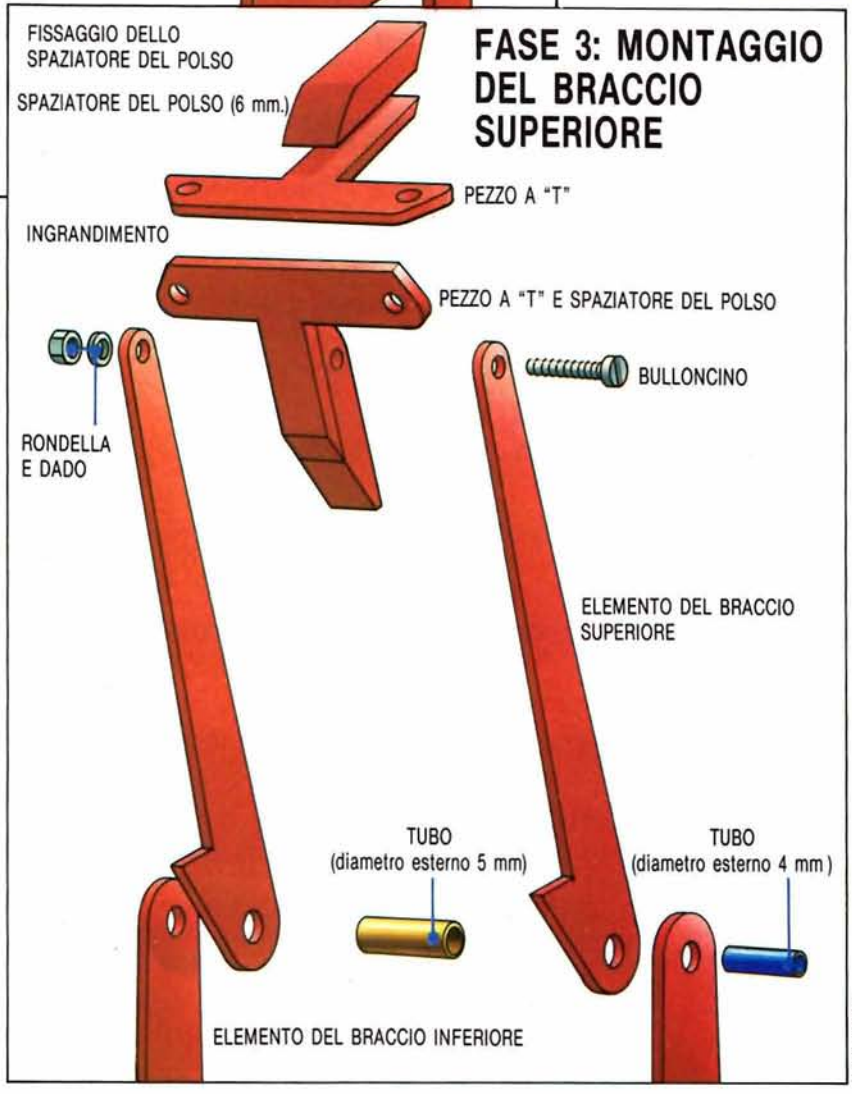
INGRANDIMENTO

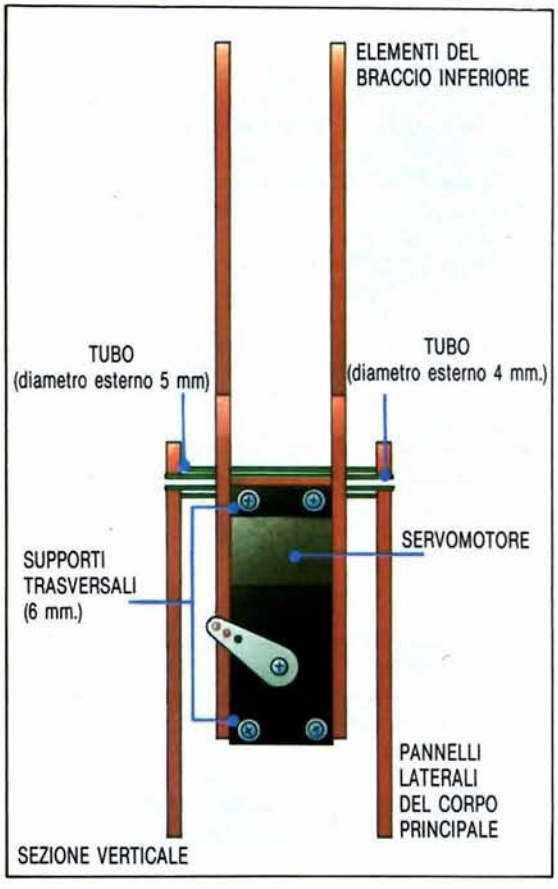






FASE 3: MONTAGGIO DEL BRACCIO SUPERIORE





KEVIN JONES

Gli insiemi nel PASCAL

Un linguaggio con molte strutture di dati: ecco gli insiemi

Spesso si parla di un insieme, o set di caratteri, di un computer. Ma cosa s'intende con precisione per "insieme" e che differenza passa tra un insieme e una matrice? Essenzialmente un insieme è una raccolta di oggetti che possono essere elaborati come un'entità unica, anziché come una serie di elementi distinti da trattare individualmente. Un esempio pratico può essere quello di tutti i programmatori che lavorano in PASCAL oppure quello delle vocali dell'alfabeto. Di solito un insieme non richiede classificazioni complicate: l'unica discriminante è se un oggetto fa parte o no di un particolare insieme.
Tuttavia il PASCAL impone alcuni vincoli sui set che, tutto sommato, ne semplificano l'impiego. Di un insieme non possono far parte matrici o altri dati strutturati ma solo scalari semplici. Inoltre la gamma consentita per questo tipo base di scalare deve possedere un valore limite superiore. Come dimostra l'esempio seguente, la sintassi PASCAL relativa alla struttura degli insiemi è semplice:

```
TYPE
   Numeri = SET OF 0..127;
   Alfabeto = SET OF 'A'..'Z';
   Colori = SET OF (Rosso, Verde, Blu);
```

Per formulare la gamma di valori consentiti dal tipo base dell'insieme, il PASCAL ricorre alla sintassi usata per i sottogruppi di dati. Come tutte le variabili PASCAL, anche quelle degli insiemi vengono definite nella sezione VAR e possono apparire all'interno di istruzioni come "letterali" racchiusi tra parentesi tonde. Ad esempio:

```
VAR
   codici : Numeri;
   tavolozza : Colori;
BEGIN
   codici := [0..2,4,8,16,32,64];
   tavolozza := [Rosso..Blu]; 'ecc.
```

Questo frammento di programma inizializza i codici dell'insieme assegnando loro solo i numeri: da 0 a 2 compreso, 4, 8, 16 ecc. Poiché, al contrario del suo tipo base, questo insieme non deve necessariamente possedere un ordine interno, i suoi elementi potrebbero essere formulati anche in modo diverso: [64, 32, 16, 8, 4, 2..0]. In questo caso, tuttavia, l'arco di valori tra 2 e 0 (non consentito in una definizione di sottoinsieme) indica semplicemente una serie di valori vuota. In realtà, qualsiasi insieme può essere inizializzato ad un insieme vuoto del suo tipo con il formato SetQualsiasi:=[]; questa è l'unica eccezione alla regola generale del PASCAL secondo la quale la semplice osservazione consente di riconoscere il tipo di qualsiasi "letterale". Infatti, se nessun elemento dell'insieme è un letterale, né il programmatore né il compilatore sono in grado di riconoscerne il tipo. Fortunatamente, l'insieme vuoto compare solo nelle istruzioni di assegnazione (come nell'esempio) oppure in espressioni nelle quali è già stato dichiarato il tipo degli altri identificatori. Ciò significa, dunque, che l'insieme vuoto è un sottoinsieme di qualsiasi insieme, ma ciò è naturale ed accettabilissimo.

Gli operatori degli insiemi

Tra i diversi operatori previsti dal PASCAL per le strutture set (ad esempio, DIV e MOD), uno dei più utili è indubbiamente la parola riservata IN. Questo operatore di tipo relazionale, che richiede la presenza di due operandi, permette di verificare la condizione di appartenenza all'insieme di un particolare elemento. L'operando di sinistra deve essere un parametro atto a valutare uno dei possibili elementi dell'insieme, ossia un valore del tipo base dell'insieme; quello di destra può essere una variabile o un letterale dell'insieme. L'espressione fornisce un risultato di tipo booleano: vero, se il valore è un elemento dell'insieme, falso, se non lo è.
Quindi N IN codici e Verde IN tavolozza sono espressioni booleane valide. L'espressione necessaria per verificare se un dato carattere è un numero, viene comunemente scritta nel seguente modo:

```
IF (c>='0') AND (c<='9') THEN...
```

In PASCAL esiste invece un'alternativa molto più concisa e chiara per verificare se il valore di c è un elemento dell'insieme (set) dei caratteri numerici:

```
IF c IN ('0'..'9') THEN...
```

Se si sta sviluppando un programma per giocare una partita a carte, può essere utile scrivere:

PRIORITÀ DEGLI OPERATORI

La tabella mostra l'ordine di priorità, alterato dall'uso di parentesi tonde che costringono a valutare separatamente l'espressione che esse racchiudono

Priorità	Operatori	Tipo
Massima	NOT + −	(unari)
	AND * / DIV MOD	(di moltiplicazione)
	OR + −	(di somma)
Minima	< <= <> >= > IN	(di tipo relazionale)

```
TYPE
    gamma = (due, tre, quattro, cinque, sei, sette, otto, nove,
        dieci, Fante, Donna, Re, Asso);
    Mazzo = SET OF gamma;
VAR
    carta : gamma;
    figure : Mazzo;
BEGIN
    figure := (Fante..Asso);
    IF carta IN figure THEN (ecc.)
```

Si confronti quest'ultima linea con l'ingombrante alternativa:

```
IF (carta = Fante) OR (carta = Donna) OR (carta = Re) OR ...
```

Il PASCAL consente anche di eseguire operazioni sugli insiemi, considerati come entità globali: queste sono l'intersezione, l'unione e la differenza. Se B è l'insieme di tutti i programmatori BASIC e P quello dei programmatori PASCAL, l'intersezione dei due insiemi è l'insieme dei programmatori che usano entrambi i linguaggi. L'unione di B e P è invece l'insieme dei programmatori che lavorano in BASIC o in PASCAL, vale a dire la combinazione dei due insiemi.
Infine la differenza è il risultato della separazione o sottrazione di tutti i membri di un insieme dall'altro; quindi, in notazione PASCAL, P−B rappresenta tutti coloro che programmano in PASCAL ma non in BASIC, P+B indica un'unione e P*B un'intersezione. Que sti operatori simbolici sono identici ai comuni operatori aritmetici, ma le operzioni che essi eseguono non vanno confuse con quelle aritmetiche. In realtà, per quanto riguarda le operazioni sugli insiemi, tali operatori possono essere paragonati alle operazioni di verifica di bit in un byte.
Si immagini un insieme composto da 8 elementi. Per indicare la presenza di un particolare elemento possiamo porre a 1 il corrispondente bit di un byte; in assenza di quell'elemento, il relativo bit viene azzerato. Poiché la verifica degli elementi dell'insieme richiede l'impiego di una maschera o "bit test", un'operazione di unione si trasforma in una OR logica ed una intersezione in una AND. Tali operazioni sono sempre eseguibili a livello di codice macchina, quindi il compilatore del PASCAL dispone di una efficace serie di strutture per i dati dell'insieme e delle operazioni loro associate. La memoria impegnata è piuttosto limitata e, quando è possibile mantenere le dimensioni degli insiemi inferiori a quelle della "parola" usata dal computer (ad esempio, su apparecchi a 32 bit o superiori), le operazioni sugli insiemi si dimostrano tra le più efficienti previste in PASCAL.

Ordine di priorità

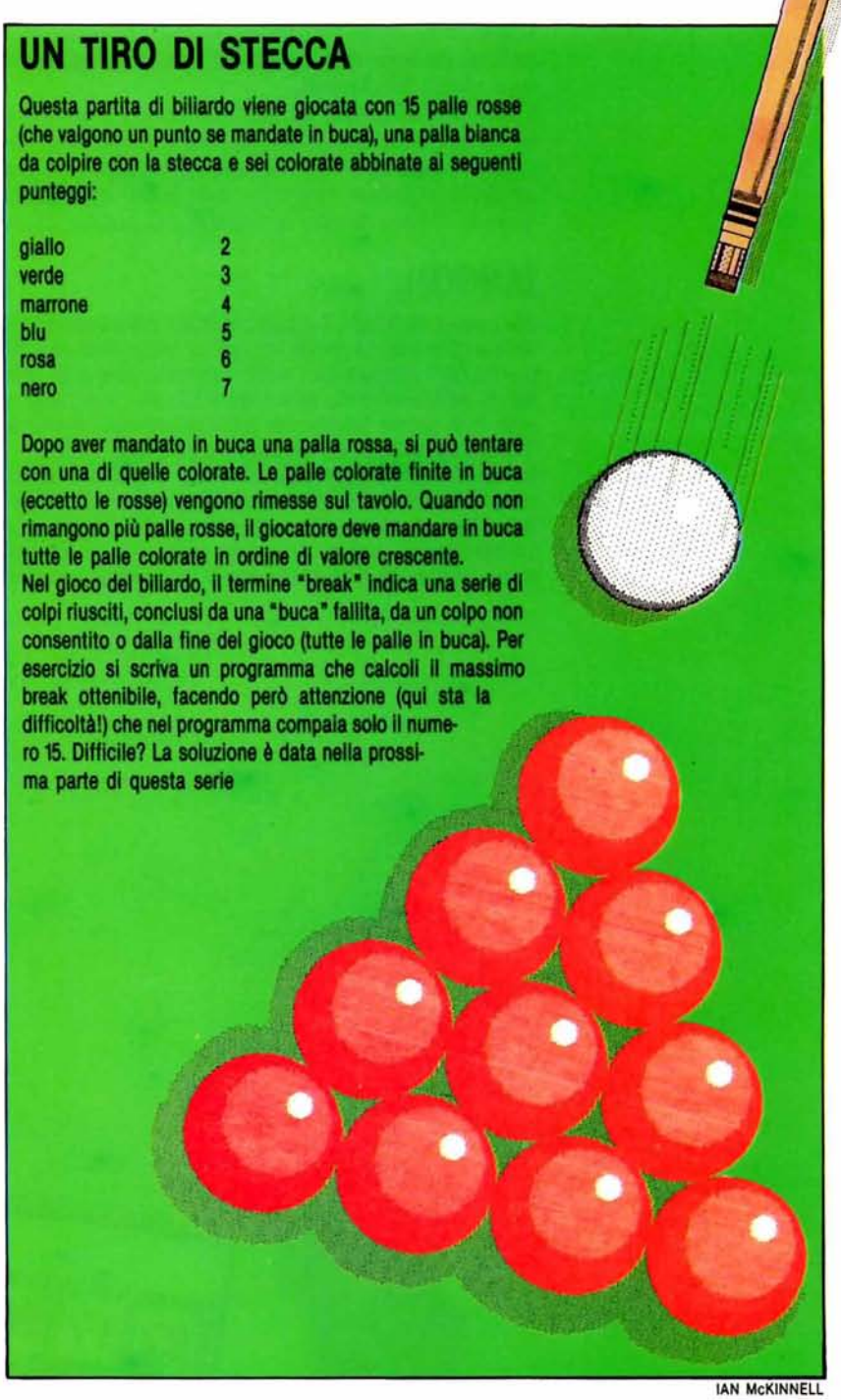
A questo punto conviene fare un breve riepilogo di tutti gli operatori del PASCAL (quelli non descritti in modo specifico si intendono comuni ad altri linguaggi). Gli operatori del PASCAL possiedono solo quattro livelli di priorità. Naturalmente i primi ad essere eseguiti sono gli operatori "monadici" (o "unari") come il + e il −, che indicano il segno di un numero, e l'operatore booleano di negazione NOT. I secondi, in ordine di priorità sono gli operatori di moltiplicazione (compresi i simboli di divisione), seguiti da quelli di addizione e sottrazione. Gli ultimi sono gli operatori di tipo relazionale tra cui IN.
Si noti che i due operatori booleani AND e OR vengono considerati rispettivamente operatori di moltiplicazione e divisione. Il loro livello di priorità segue quindi le regole dell'algebra booleana e comporta l'inserimento tra parentesi tonde di numerose verifiche di tipo relazionale per modificarne la priorità. In caso contrario, l'istruzione IF N>0 AND N<10 THEN... produce un "time error" in fase di compilazione perché l'espressione 0 AND N, che per la

UN TIRO DI STECCA

Questa partita di biliardo viene giocata con 15 palle rosse (che valgono un punto se mandate in buca), una palla bianca da colpire con la stecca e sei colorate abbinate ai seguenti punteggi:

giallo	2
verde	3
marrone	4
blu	5
rosa	6
nero	7

Dopo aver mandato in buca una palla rossa, si può tentare con una di quelle colorate. Le palle colorate finite in buca (eccetto le rosse) vengono rimesse sul tavolo. Quando non rimangono più palle rosse, il giocatore deve mandare in buca tutte le palle colorate in ordine di valore crescente.
Nel gioco del biliardo, il termine "break" indica una serie di colpi riusciti, conclusi da una "buca" fallita, da un colpo non consentito o dalla fine del gioco (tutte le palle in buca). Per esercizio si scriva un programma che calcoli il massimo break ottenibile, facendo però attenzione (qui sta la difficoltà!) che nel programma compaia solo il numero 15. Difficile? La soluzione è data nella prossima parte di questa serie



IAN McKINNELL

mancanza delle parentesi viene valutata per prima, collega due operandi interi tramite un operatore booleano.
Come d'abitudine, gli operatori con pari livello di priorità vengono valutati a partire da sinistra. Il simbolo di assegnazione := ha priorità minore rispetto agli altri operatori quindi l'espressione a destra del segno deve essere valutata prima di eseguire l'assegnazione. Un'ultima raccomandazione: non conviene mai supporre che un'espressione non venga valutata in tutte le sue possibilità; ad esempio, un'espressione del tipo: IF (N>0) AND (K/N<10) THEN... può generare un errore di divisione per zero, se N vale 0, e provocare così la "caduta" di un programma.

BINGO!!

Una cartella del Bingo o Tombola può essere facilmente rappresentata da un insieme. Sebbene gli elementi base siano ordinati (numeri interi compresi tra 1 e 90), il giocatore esegue un solo tipo di verifica: controlla se il numero chiamato è presente o meno sulla cartella. Questa corrisponde all'operazione booleana "numero IN cartella", che dà come risultati "vero" o "falso".
Il programma simula il gioco della Tombola: innanzi tutto legge i numeri della cartella inseriti da tastiera; quindi, ad ogni "estratto", verifica se questo è contenuto nella cartella. L'espressione Cartella-Estratto produce un insieme vuoto quando tutti gli elementi di Cartella si trovano anche nell'insieme Estratto, ossia quando si è fatta "tombola". L'introduzione di un elemento in un insieme che già lo contiene, non modifica l'insieme, così come rimuovere un elemento non appartenente all'insieme. Si noti che non è consentito aggiungere direttamente un elemento ad un insieme, ma l'ostacolo si aggira creando un insieme composto da tale elemento, ossia racchiudendo il numero tra parentesi quadre, per poi operare l'unione di due insiemi.
Il programma consente di creare anche cartelle duplicate e accetta l'immissione di numeri non compresi tra 1 e 90 provocando, in quest'ultimo caso, un "run-time error". Come esercizio inserire un ciclo che elimini queste anomalie e rifiuti gli estratti

```
PROGRAMMA               Bingo          (input, output);

CONST
          Colonne = 40;                 {variare a seconda dello
                                         schermo}
          PosCentro = 25;               {variare a seconda dello
                                         schermo}

TYPE
          Bingo         = SET OF 1 .. 90;

VAR       conto         : 1 .. 15;
          Gamma,
          Estratto,
          Vuoto,
          Cartella      : Bingo;
          numero        : intero;
          Tombola       : booleana;

BEGIN
    Vuoto := [ ]
    Gamma := [1 .. 90];
    WriteLn ('*** BINGO ***': PosCentro);
    WriteLn;
    WriteLn ('Inserire i 15 numeri della cartella,');
    WriteLn ('(ciascuno seguito da RETURN) :');
    Cartella := Vuoto;

    FOR conto := 1 TO 15 DO
        BEGIN
        write (conto : 10, ': ? ' );
        ReadLn (numero);
        Cartella := Cartella + [numero]
END;

WriteLn;
WriteLn ('NON SBIRCIARE !' : PosCentro);
WriteLn;
WriteLn ('Comincia a estrarre i numeri : ');
Estratto := Vuoto;

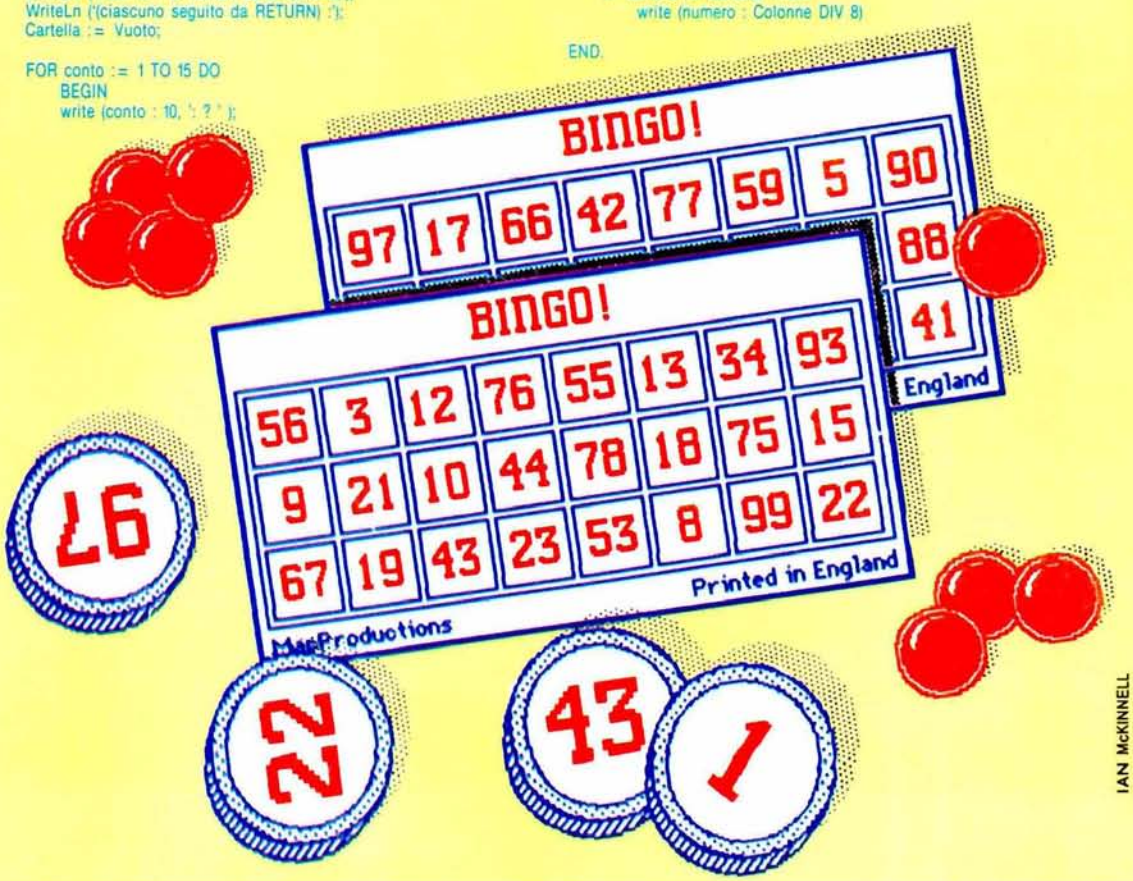
REPEAT
    write ('? ' : PosCentro);
    ReadLn (numero);
    Estratto := Estratto + [numero];
    Tombola := Cartella — Estratto = Vuoto

UNTIL Tombola;

WriteLn;
WriteLn ('Congratulazioni !' : PosCentro);
WriteLn ('I tuoi numeri erano :');
WriteLn;

FOR numero := 1 TO 90 DO
    IF numero IN Cartella THEN
        write (numero : Colonne DIV 8)

END.
```



IAN McKINNELL

Ultima chiamata

Termina la serie dedicata al SO del Micro BBC

Il programma dimostrativo descritto in precedenza (vedere a pag. 1580) dimostra come sia possibile intercettare uno dei vettori del SO, al fine di cambiare la risposta del SO ad un particolare evento. Il programma modifica il contenuto di OSWRCHV, il vettore che contiene l'indirizzo della routine OSWRCH. Questo vettore si trova agli indirizzi $20E e &20F.
Una delle funzioni di OSWRCH è prendere in carico il testo di un programma BASIC e visualizzarlo sullo schermo quando il programma viene listato. Nel nostro programma, la funzione principale del codice macchina è esaminare il codice ASCII di tutti i caratteri che passano attraverso l'accumulatore durante la routine OSWRCH. Se il codice ASCII è compreso tra 64 e 91, il carattere corrispondente è una lettera maiuscola e la routine somma al suo valore il numero 32 per trasformarlo nella minuscola equivalente. Quindi il programma è semplicemente un "cuneo" che viene eseguito tutte le volte che il sistema operativo chiama OSWRCH. Una volta eseguito il codice, un'istruzione JMP seguita dal vecchio contenuto di OSWRCHV trasferisce il controllo alla vera routine OSWRCH. Al termine del programma, tutte le lettere maiuscole del programma listato sullo schermo o inserite da tastiera risultano trasformate in minuscole dalla routine cuneo. L'uso del tasto BREAK ripristina in OSWRCHV il suo valore originale.

Programma per l'uso della memoria

Il listato a fianco è un altro programma di utilità, che controlla l'area di memoria occupata durante la programmazione. L'espressione:

HIMEM–(?2+256*?3)

comunica la quantità di memoria non utilizzata dopo l'immissione del programma e delle variabili. Le locazioni 2 e 3 contengono l'indirizzo dell'estremità superiore della tabella per le variabili BASIC, quindi l'espressione indica quanta memoria rimane a disposizione dell'utente. Tuttavia, in alternativa alla ripetizione dell'istruzione PRINT, per visualizzare questo valore, la routine usa l'evento Introduzione di un carattere nel buffer d'ingresso (vedere a pag. 1579) per avviare il calcolo dell'espressione, ogni volta che viene premuto un tasto. Quando la memoria BASIC disponibile cala sotto un livello prefissato, la routine emette un "bip". La routine continua a lavorare in sottofondo, mentre l'utente digita tranquillamente il proprio programma.
Per prima cosa il programma chiede all'utente di indicare il valore di soglia, al di sotto del quale deve comunicare le residue disponibilità di memoria. Questa operazione viene svolta dalla procedura PROCvalore–soglia, che definisce il valore soglia nel formato byte basso/byte alto (lo-hi). Il programma viene collocato all'indirizzo &0A00 e, come d'abitudine nei programmi che usano gli eventi, si deve salvare sullo stack il contenuto di tutti i registri. Eseguita la routine di salvataggio, il programma azzera i registri ed esegue un'istruzione RTS che riporta il controllo del programma dove si trovava quando si è verificato l'evento.
La routine principale chiama altre due subroutine, che servono per valutare la quantità di memoria residua. La routine inc somma il valore contenuto nelle locazioni 2 e 3 ai byte basso e alto della soglia e deposita il risultato

USO DELLA MEMORIA

```
100 REM **** Evento generato da tastiera ****
110 REM **** Osservazione dell'uso della memoria ****
120 MODE 1
130 PROCvalore_soglia
140 PROCassembla
150 END
160
170 DEF PROCvalore_soglia
180 INPUT "Emissione di un "bip" per num. di Kbyte
    ancora liberi",n: n=n*1024
190 hibyte=n DIV 256
200 lobyte=n MOD 256
210 ENDPROC
220
230 DEF PROCassembla
240 oswrch=&FFEE
250
260
270 FOR pass=0 TO 3 STEP 3
280 P%=&0A00
290
300 [OPT pass
310 .inizio
320 PHP
330 PHA
340 TXA:PHA
350 TYA:PHA
360
370 JSR verifica_memoria
380
390
400 PLA:TAY
410 PLA:TAX
420 PLA
430 PLP
440 RTS
450
460 .verifica_memoria
470 JSR inc
480 JSR sub
490 BPL room
500 LDA #7
510 JSR oswrch
520 .room RTS
530
540 .sub SEC
550 LDA &06:SBC &70:STA &70
560 LDA &07:SBC &71:STA &71
570 RTS
580 .inc CLC
590 LDA &02:ADC #lobyte:STA &70
600 LDA &03:ADC #hibyte:STA &71
610 RTS
620
630 ]:NEXT
640 ?&220=start MOD 256
650 ?&221=start DIV 256
660 *FX14,2
670 ENDPROC
```

nelle locazioni &70 e &71. La routine sub esegue un calcolo a 16 bit, sottraendo da HIMEM il contenuto di &70 e &71. Il valore di HIMEM è contenuto dalle locazioni di pagina zero &06 e &07. Si noti che il programma usa locazioni di memoria assolute, poiché il loro indirizzo è comune a tutte le versioni del BASIC BBC.
Dopo la sottrazione, la linea 490 verifica il segno del risultato: se è positivo, il numero dei byte di memoria ancora disponibili è maggiore di quello indicato dal valore soglia. In caso contrario la memoria residua è inferiore al valore soglia e viene pertanto eseguita la chiamata OSWRCH con A=7 che produce l'emissione del "bip".
I due programmi sin qui descritti sono semplici esempi di applicazioni dalle notevoli potenzialità. Uno dei vantaggi delle chiamate dirette al SO, funzioni altrimenti non disponibili in BASIC, è la possibilità di creare un codice pilotato da eventi, come in entrambi i programmi seguenti.

Sottofondo musicale

Questo programma adopera l'evento Tempo scaduto del temporizzatore d'intervalli per suonare una serie di note musicali, mentre il computer esegue altri lavori. Poiché la routine dell'evento viene eseguita in sottofondo, è possibile salvare un programma, editarlo, listarlo oppure eseguire un diverso programma anche durante l'emissione delle note. Volendo, si potrebbe ampliare il programma per suonare una canzone durante l'esecuzione di un programma di gioco (la musica viene suonata anche durante gli accessi ai dischetti). Il programma viene assemblato a partire dall'indirizzo &0C00. La prima sezione è una routine di inizializzazione che esegue le seguenti operazioni: modifica il vettore dell'evento EVENTV, registrato agli indirizzi $220 e $221, affinché punti alla nostra routine; chiama OSBYTE 14 con X=5 per attivare l'evento Tempo scaduto; azzera un contatore di note posto alla locazione &70; infine chiama la subroutine clock, che avvia il temporizzatore di eventi.
Come prima operazione, la routine dell'evento salva sullo stack i registri della CPU. Quindi chiama una subroutine per suonare una sola nota e azzera l'orologio del temporizzatore. Infine ripristina i registri. Si noti che le sezioni del programma per il salvataggio e il ripristino dei registri sono pressoché identiche in tutte le routine di gestione dell'evento: cambia soltanto il codice inserito tra questi due blocchi d'istruzioni.
Esaminiamo adesso le diverse parti della subroutine che genera le note. Queste vengono selezionate da una tabella di note e suonate ogni volta che il temporizzatore di eventi raggiunge lo zero. La tabella, chamata tabnote è composta da una serie di byte contenenti l'informazione d'altezza per le 5 note che vogliamo suonare. La subroutine usa un contatore, posto all'indirizzo &70, per indicare quale delle 5 note deve essere suonata. Il valore dell'altezza viene estratto e registrato nell'appropriata posizione all'interno di tabsuono, ossia il blocco dei parametri per la routine OSWORD, usata più avanti per generare il suono. Quindi viene incrementato il contatore per puntarlo alla posizione seguente della tabella delle note: il valore dell'altezza in essa contenuto viene usato dalla successiva esecuzione della routine. Poiché questa tabella contiene solo 5 note, il contatore, in &70, viene azzerato non appena supera il valore 5.
La subroutine clock usa OSWORD con A=4 per avviare il temporizzatore dell'evento. L'area di memoria chiamata tempo contiene il valore da immagazzinare nei registri del temporizzatore, fissando quindi l'intervallo tra le note suonate. La subroutine suono usa la chiamata OSWORD con A=7 per produrre lo stesso risultato di un comando SOUND; tabsuono rappresenta il relativo blocco dei parametri. I dati per tabnote e tabsuono vengono forniti dalle istruzioni BASIC alle linee 840 e 850.
Il tasto BREAK interrompe il programma, ripristinando il contenuto originale del vettore EVENTV, mentre una nuova chiamata alla routine inizializza fa ripartire la musica.

SOTTOFONDO MUSICALE

```
 10 REM **** EVENTI DI SUONO ****
 50
 60 PROCassembla
 80 CALL inizializza
 90 END
100
110 DEF PROCassembla
120 cont_note=&70
130 oswrch=&FFEE
140 osbyte=&FFF4
150 osword=&FFF1
160
170 FOR pass=0 TO 2 STEP 2
180 P%=&0C00
190
200 [OPT pass
210 .inizializza
220 LDA #event MOD 256:STA &220
230 LDA #event DIV 256:STA &221
240 LDA #14:LDX #5:JSR osbyte
250 LDA #0:STA cont_note
260 JSR clock
270 RTS
280
290 .evento
300 PHP
310 PHA
320 TXA:PHA
330 TYA:PHA
340
350 JSR suona_nota
360 JSR clock
370
380
390 PLA:TAY
400 PLA:TAX
410 PLA
420 PLP
430 RTS
440
450 .suona_nota
460 LDY cont_note
470 .sciclo LDA tabnote,Y
480 STA tabsuono+4
490 JSR suono
500 INC cont_note
510 LDA cont_note: CMP#5: BNE out
520 LDA #0: STA cont_note
530 .out RTS
540
550
560 .clock
570 LDX #time MOD 256
580 LDY #time DIV 256
590 LDA #4
600 JSR osword
610 RTS
620
630 .suono PHA
640 TXA:PHA
650 TYA:PHA
660 LDX #tabsuono MOD 256
670 LDY #tabsuono DIV 256
680 LDA #7
690 JSR osword
700 PLA:TAY
710 PLA:TAX
720 PLA
730 RTS
740
750 .tabnote
760 EQUD &00000000
770 EQUD &00000000
780 .tabsuono
790 EQUD &00000000
800 EQUD &00000000
810 .tempo EQUD &FFFFFFCF
820 EQUB &FF
830 ]:NEXT
840 FOR I%=0 TO 4: READ data:?
    (tabnote+I%)=data: NEXT
850 FOR I%=0 TO 7: READ data:?
    (tabsuono+I%)=data: NEXT
860 ENDPROC
870 DATA 69,73,81,89,97
880 DATA 1,0,&FA,&FF,0,0,10,0
```

Grafica in movimento

Quest'ultimo programma mostra come sia possibile usare gli eventi per muovere sullo schermo una semplice figura geometrica, in questo caso un rettangolo, cambiandone la posizione ad ogni evento. La frequenza del movimento dipende dal valore introdotto nel temporizzatore di evento e specificato alla linea 1260. La figura che deve essere mossa, definita in termini di valori numerici PLOT, viene immagazzinata in tabfigura e creata da un'istruzione DATA: la linea 1220 inserisce (POKE) i valori appropriati. Le linee 1230-1250 modifica-

GRAFICA IN MOVIMENTO

```
10 REM **** GRAFICA & TEMPORIZZATORE DI EVENTI ****
20 MODE 1
40 PROCassembla
50 CALL clock
60 END
70 DEF PROCassembla
80 xpos=&70:ypos=&72
90 ?xpos=00:?(xpos+1)=0
100 ?ypos=100:?(ypos+1)=0
110 oswrch=&FFEE
120 osbyte=&FFF4
130 osword=&FFF1
140 DIM codice% 600, tempo% 10
150
160 FOR pass=0 TO 2 STEP 2
170 P%=codice%
180
190 [OPT pass
200 PHP
210 PHA
220 TXA:PHA
230 TYA:PHA
240
250 JSR salva
260 JSR traccia
270 JSR ripristina
280 JSR clock
290
300
310 PLA:TAY
320 PLA:TAX
330 PLA
340 PLP
350 RTS
360
370 .salva
380 LDX #pgpos MOD 256
390 LDY #pgpos DIV 256
400 LDA #&0D
410 JSR osword
420 LDA &35B:STA ccol
430 LDA &35C:STA ccol+1
440 RTS
450
460 .traccia
470 JSR gcol
480 JSR muove
490 JSR figura
500 JSR incx
510 JSR incy
520 RTS
530
540 .ripristina
550 LDA ccol:STA&35B
560 LDA ccol+1:STA&35C
570 LDA #25:JSR oswrch
580 LDA #4:JSR oswrch
590 LDA cgpos:JSR oswrch
600 LDA cgpos+1:JSR oswrch
610 LDA cgpos+2:JSR oswrch
620 LDA cgpos+3:JSR oswrch
630 RTS
640
650 .muove
660 LDA #25:JSR oswrch
670 LDA #4:JSR oswrch
680 LDA xpos:JSR oswrch
690 LDA xpos+1:JSR oswrch
700 LDA xpos+2:JSR oswrch
710 LDA xpos+3:JSR oswrch
720 RTS
730
740 .figura
750 LDX#24:LDY#0
760 .sciclo LDA tabfigura,Y
770 JSR oswrch
780 INY:DEX
790 BNE sciclo
800 RTS
810
820 .incx CLC
830 LDA #2:ADC &70:STA &70
840 LDA #0:ADC &71:STA &71
850 RTS
860 .incy CLC
870 LDA #2:ADC &72:STA &72
880 LDA #0:ADC &73:STA &73
890 LDA &73: CMP #3: BCS overflow
900 RTS
910 .overflow
920 LDA #0:STA &72:STA &73
930 LDA #0:STA &70:STA &71
940 RTS
950
960 .clock
970 LDX #tempo% MOD 256
980 LDY #tempo% DIV 256
990 LDA #4
1000 JSR osword
1010 RTS
1020
1030
1040 .gcol
1050 LDA #18:JSR oswrch
1060 LDA #3:JSR oswrch
1070 LDA #1:JSR oswrch
1080 RTS
1090
1100
1110 .ccol EQUD &0000
1120 .pgpos EQUD &00000000
1130 .cgpos EQUD &00000000
1140 .tabfigura
1150 EQUD &00000000
1160 EQUD &00000000
1170 EQUD &00000000
1180 EQUD &00000000
1190 EQUD &00000000
1200 EQUD &00000000
1210 ]:NEXT
1220 FOR I%=0 TO 23: READ data:?
     (tabfigura+I%)=data: NEXT
1230 ?&220=codice% MOD 256
1240 ?&221=codice% DIV 256
1250 *FX14,5
1260 !tempo%=&FFFFFFEF:
     tempo%?4=&FF
1270 ENDPROC
1280 DATA 25,1,60,0,0,0,25,1,0,0,
     60,0,25,1,&C4,255,0,0,25,1,0,
     0,&C4,255
```

no il vettore EVENTV, affinché punti all'inizio del programma e abilitano l'evento. Esaminiamo adesso le parti in codice macchina del programma.

Il codice macchina viene immagazzinato in memoria all'indirizzo indicato da CODICE%. Le linee 200-230 salvano, come di consueto, i vari registri mentre le linee 250-280 chiamano una serie di subroutine che eseguono le operazioni necessarie. Infine vengono ripristinati i registri ed eseguita un'istruzione RTS. Le linee 370-440 salvano lo stato corrente del cursore grafico e il GCOL in uso che può così essere ripristinato dopo aver gestito l'evento. Le linee 460-520 controllano il movimento della figura. Le linee 540-630 riportano i colori della grafica e la posizione del cursore grafico allo stato precedente al lancio della routine. Le linee 650-720 usano la routine OSWRCH come equivalente del comando BASIC MOVE per specificare le posizioni di x e y: quelle della coordinata x vengono registrate in xpos e xpos+1, quelle di y in ypos e ypos+1. Le linee 740-800 adoperano nuovamente OSWRCH per tracciare il rettangolo. Il registro Y viene usato come un registro d'indice per avere accesso alla tabella per la figura (tabfigura): i byte che la compongono vengono trasmessi in sequenza tramite OSWRCH.

Poiché la figura si muove sullo schermo, ci devono essere delle routine che aggiornano le posizioni di x e y ogni volta che si verifica un evento. Tale operazione viene eseguita dalle routine incx e incy (linee 820-900). Quando il rettangolo raggiunge il margine superiore dello schermo, la routine overflow azzera le coordinate di x e y. La routine clock usa la chiamata OSWORD con A=4 per azzerare il temporizzatore di eventi mentre la routine gcol esegue la stessa operazione del comando BASIC GCOL 3,1.

Questo programma, però, può dare origine a numerosi inconvenienti: durante la sua esecuzione, finché lo schermo non esegue uno scroll, la figura si muove in modo incerto su di esso. In questa fase è possibile eseguire altre operazioni senza incontrare difficoltà. Tuttavia, se acceleriamo il movimento, incrementando il valore nell'area di memoria tempo, e contemporaneamente visualizziamo il listato di una parte del programma, la figura continua a muoversi ma sullo schermo possono comparire caratteri non previsti. La causa di questo difetto è evidente: quando visualizziamo un listato, la routine chiamata al verificarsi dell'evento entra in conflitto con la routine del SO che visualizza il listato. Poiché entrambe usano OSWRCH, quando l'evento interrompe la routine del SO, OSWRCH viene lasciata in uno stato che non viene poi recuperato.

Se OSWRCH viene interrotta e subito dopo chiamata dalla routine d'interruzione, non è più impossibile uscire dalla chiamata di OSWRCH eseguita dalla routine d'interruzione. La routine OSWRCH viene detta "non rientrante". Per tale motivo la Acorn sconsiglia di usare le routine del SO all'interno di routine per la gestione delle interrupt e degli eventi. Il loro impiego non è vietato, ma occorre procedere con grande cautela. Una delle alternative, per tracciare figure grafiche, può essere l'inserimento diretto dei dati nell'area RAM per lo schermo.

Software in vetrina

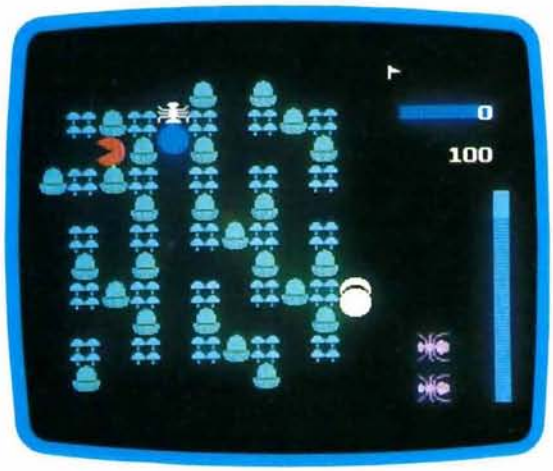
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

È la volta di due giochi per lo Spectrum della Sinclair, ricchi di azione e movimento: "olimpionico" il primo, vero "arcade" il secondo.

"Daley's Decathlon"

Obiettivo del gioco, facile da intuire con un simile titolo, è quello di portare a termine con successo dieci prove atletiche (nella finzione s'intende!), cercando di totalizzare il maggior numero di punti. Nelle gare di movimento l'azione è controllata tramite la pressione alternata di due tasti oppure per mezzo di un joystick. La grafica è molto realistica e il pubblico, dalle tribune, esulta rumorosamente ad ogni tentativo coronato da successo. Il gioco è prodotto dalla Ocean (GB).





"Hyperaction"

La Silversoft (GB) propone una nuova versione di un gioco che appartiene a un filone tra i più classici per i videogame: quello dei vari Pacman. Tuttavia, *Hyperaction* offre un'interessante novità. Il gioco, infatti, si svolge a più livelli: quando si riesce a superare il primo labirinto ne viene proposto uno in cui occorre, oltretutto, adottare una tattica di gioco diversa. Salendo di livello aumenta anche la difficoltà: occorrono sempre più astuzia e rapidità per sfuggire ai vari Pacman che rincorrono il protagonista.

Linguaggi moderni

Domande e risposte sull'informatica nella scuola

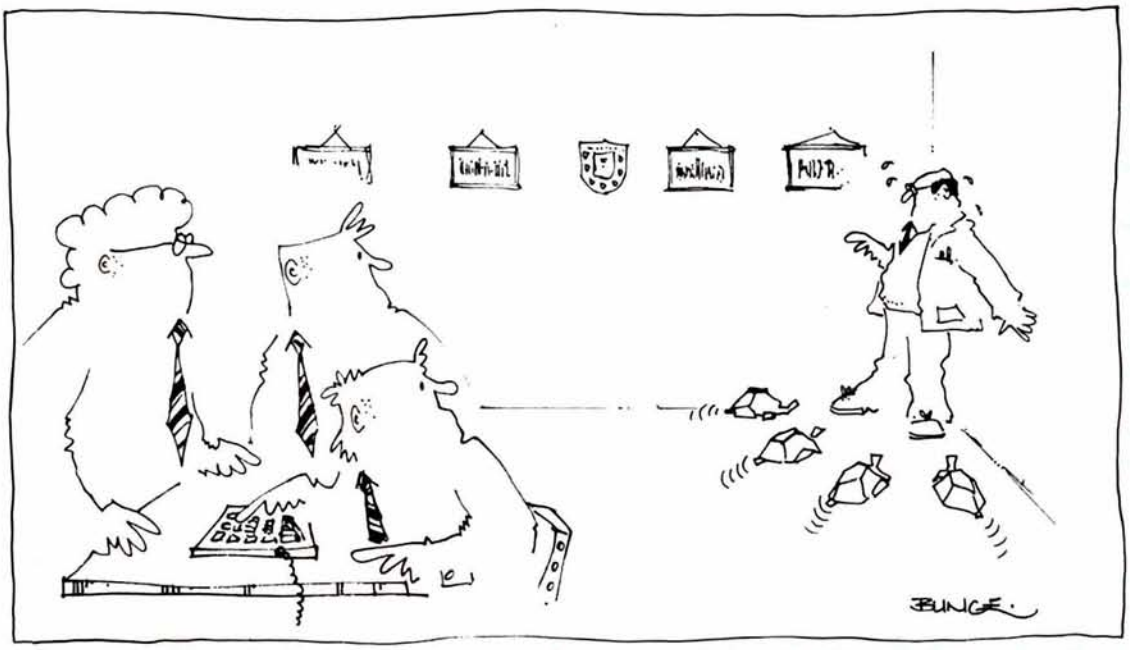
D. Perché esistono notevoli controversie sull'insegnamento della programmazione del computer nelle scuole primarie?

R. Il ruolo tradizionale dell'allievo è quello di "passivo ricettacolo", in cui riversare il "sapere". Se, al contrario, l'allievo assume un ruolo attivo si assiste ad un inevitabile sconvolgimento. Infatti per quanto riguarda ad esempio l'insegnamento dell'informatica, non esiste ancora un valido criterio da seguire, con conseguenti difficoltà sia da parte degli insegnanti che degli allievi.
In molti paesi l'informatica fa parte già da anni dei programmi scolastici, ma se da un lato genitori e studenti approvano con entusiasmo l'introduzione di tale materia, dall'altro esiste una resistenza o diffidenza da parte di insegnanti e di coloro che operano nell'industria (sempre a caccia di nuove reclute).
Molti insegnanti affermano che l'uso dei computer non dovrebbe essere limitato al laboratorio informatico, bensì esteso a tutte le materie del programma scolastico. Tuttavia, alcune recenti esperienze hanno dimostrato che, ampliando l'impiego del computer, si rischia di sottovalutare l'insegnamento della programmazione, mentre invece gli esperti sostengono l'enorme importanza di questa materia, che abitua ad organizzare i propri pensieri in forma strutturata.
Come si è già visto in precedenza, si tratta soltanto di formare una cultura di base. Ci sono materie come la storia dell'arte, che vengono insegnate anche a chi non intende diventare pittore o scultore: lo stesso dovrebbe accadere con l'informatica.
Un'altra causa di critiche è il fatto che, particolarmente agli inizi, i pochi computer in dotazione alle scuole funzionavano solo in BASIC. Questo linguaggio, per la sua stessa struttura, è troppo concettuale per bambini



Una varietà di linguaggi

L'implementazione di un numero sempre maggiore di linguaggi sui microcomputer sta lentamente ampliando le basi dello studio della programmazione nelle scuole. Lo studio di alcuni linguaggi, quale il PROLOG, può anche essere semplicemente limitato alla loro struttura. Altri linguaggi, invece, costituiscono una valida alternativa al BASIC. Quest'ultimo è facile da imparare, ma provoca la formulazione di programmi molto disordinati e privi di una buona struttura logica



al di sotto dei 12 anni e, inoltre, non è visto di buon occhio dagli esperti, in quanto scarsamente istruttivo ai fini della programmazione, soprattutto perché non è un linguaggio molto strutturato. Ciò favorisce l'instaurarsi, negli allievi, di cattive abitudini di programmazione. Sfortunatamente, molti degli insegnanti incaricati di tenere questi elementari corsi di informatica sono stati a loro volta introdotti alla programmazione passando attraverso il BASIC, quindi tendono a mantenere questa situazione nonostante siano ampiamente disponibili linguaggi migliori (LOGO, COMAL, PROLOG e via dicendo).

D. Se, come viene affermato, il BASIC porta a pessime e scorrette abitudini di programmazione, quali sono le alternative?

R. I tentativi di introdurre dialetti BASIC leggermente più strutturati (ad esempio quello del Micro BBC) sono da molti considerati come semplici espedienti commerciali. I linguaggi più adatti, infatti, sono e restano i tre appena citati. Del LOGO si è trattato ampiamente in precedenza. Questo linguaggio ha incontrato un grandissimo successo e le esperienze statunitensi e britanniche confermano la sua validità. Considerato ciò, quasi tutti i produttori di computer hanno realizzato una versione di LOGO per i propri apparecchi, facendolo così uscire dai laboratori di Intelligenza Artificiale, dove altrimenti sembrava dover rimanere confinato.
Il COMAL, il cui nome deriva da COMmon Algorithmic Language, è un linguaggio sviluppato in Danimarca da Borge Christensen, il quale afferma che si tratta di uno sviluppo strutturato del BASIC. Christensen e i suoi collaboratori trovavano i programmi redatti dagli studenti così caotici da risultare incomprensibili; da ciò l'idea di sviluppare questo nuovo linguaggio. Le tesi a sostegno del COMAL sono risultate talmente solide che il governo danese lo ha scelto come primo linguaggio da insegnare nelle scuole inferiori. Il COMAL va rapidamente diffondendosi anche in altre nazioni europee.
Il PROLOG (sigla per PROgramming in LOGic) è un linguaggio adoperato da un ristretto gruppo di insegnanti in Gran Bretagna. La sua funzione principale, in effetti, è quella di insegnare a formulare con chiarezza e coerenza i problemi da risolvere col computer, piuttosto che esercitare la pratica della programmazione in sé. Alla luce delle esperienze condotte finora, questo linguaggio è particolarmente adatto a ragazzi di età compresa tra i 10 e i 13 anni.

D. Quali sono le differenze, tra questi linguaggi, che possono influire sull'apprendimento dell'informatica da parte di un bambino?

R. Il LOGO è stato sottoposto a molteplici verifiche per stabilire fino a che punto la programmazione in tale linguaggio possa effettivamente giovare a bambini molto piccoli (purtroppo, lo stesso non si può dire, almeno per il momento, del COMAL e del PROLOG, considerata la loro limitata applicazione). Negli USA è stato messo a punto un progetto, chiamato Brookline LOGO project, per diffondere l'apprendimento del LOGO. La prima fase del progetto, in cui erano impegnati ragazzi di circa 11 anni, era costituita da attività svolte in laboratori di informatica. La seconda fase prevedeva la graduale introduzione dei computer nelle scuole e riguardava parzialmente lo sviluppo di materie scolastiche. Un'interessante particolarità, osservata durante la prima fase, è che tutti i ragazzi coinvolti ottenevano notevoli successi nel loro lavoro in LOGO, indipendentemente dal risultato nelle altre materie. Questo dato è stato osservato anche altrove, sempre nell'ambito di altri progetti con il LOGO.

D. Quali altri tentativi sono stati fatti per studiare gli effetti su bambini che imparano a programmare?

R. La New York Academy of Sciences e tre scuole di New York hanno realizzato un progetto per fornire l'addestramento e il supporto tecnico necessari a diffondere l'insegnamento del LOGO nelle scuole. Settimanalmente, un incaricato del progetto visitava le classi scolastiche in cui erano stati installati computer e gli insegnanti, volontariamente, frequentavano seminari mensili di aggiornamento sulla materia. A conti fatti, si è potuto stabilire che il progetto ha ottenuto successo nel 95% dei casi. Il direttore del progetto, Michael Tempel, ha in seguito evidenziato i molti punti in comune con il progetto Brookline. Per prima cosa è stata notata una forte e positiva interazione tra studenti, laddove il LOGO veniva applicato in classe. Anche in questo caso, i successi conseguiti dagli studenti, anche se non particolarmente brillanti in altre materie, è stato notevole. Secondo il parere di Tempel, del resto confermato dai risultati dello stesso progetto, i ragazzi rendono bene perché si trovano accomunati in una nuova esperienza, che stimola e modifica sensibilmente sia i loro rapporti sociali, sia il loro atteggiamento individuale nei confronti dell'apprendimento. Il suggerimento che scaturisce direttamente da quanto osservato durante il progetto è quello di istituire una scuola "pilota" con la funzione di guida e centro di ricerca per ogni area territoriale, a cui gli istituti limitrofi possano far riferimento.
Un ulteriore progetto, protrattosi per 30 mesi, è stato realizzato dal Center for Children and Technology di New York. Esso mirava a stabilire se e in quale misura fosse possibile trasferire i concetti, appresi durante lo studio del LOGO, ad altre materie scolastiche. Saper pianificare è considerata un'abilità molto importante nello sviluppo di un individuo e, generalmente, tale concetto è scarsamente sviluppato nei bambini. La struttura procedurale del LOGO, abituando a suddividere i problemi in una serie di elementi più controllabili e facilmente risolvibili, riesce a stimolare e a organizzare proficuamente gli sforzi di pianificazione degli allievi. L'effetto della programmazione su questo processo evolutivo è stato uno dei principali obiettivi del progetto.

D. Quali problemi hanno gravato sull'efficacia di questi studi?

R. I problemi che si incontrano nella ricerca sull'utilità di insegnare la programmazione a scuola sono numerosissimi. In primo luogo gli organi pubblici sono riluttanti a introdurre l'informatica tra le altre materie di studio, poiché, viene detto, non ne è stata ancora provata a fondo l'effettiva utilità. Inoltre c'è una notevole scarsità di fondi e, pertanto, di hardware: ciò rende difficoltoso qualsiasi tentativo di realizzare strutture didattiche realmente efficienti. In ultimo luogo, c'è da considerare la formazione (o per meglio dire la mancanza di formazione) degli insegnanti che, molto spesso, sono digiuni in materia di informatica quanto o forse più degli stessi studenti.

D. Oltre allo sviluppo di una migliore logica di lavoro, quali altre applicazioni pratiche possono derivare, per un bambino, dall'apprendimento della programmazione?

R. Una volta introdotto nel mondo della programmazione, il numero di potenziali applicazioni che si presentano ad un bambino è enorme, limitato soltanto dalla fantasia. Ad esempio, un gruppo di bambini ha ideato un programma coreografico, creando un vero e proprio corpo di ballo composto da tartarughe pilotate dal LOGO. Lo stesso linguaggio viene frequentemente adoperato, in classi sperimentali, per redigere componimenti e racconti: l'allievo, al quale il computer presenta le possibili diramazioni della trama, sceglie come condurre lo svolgimento del racconto in base a semplici risposte digitate sulla tastiera. Inoltre, poiché alcune versioni di LOGO prevedono comandi adeguati, gli allievi possono dedicarsi anche alla composizione di semplici brani musicali o limitarsi alla esecuzione di accordi.

D. Se l'insegnamento della programmazione è limitato a poche scuole e, inoltre, vengono adoperati linguaggi diversi, com'è possibile ottenere buoni risultati su larga scala?

R. Prendendo nuovamente ad esempio la Gran Bretagna, dove l'informatica a livello infantile si è largamente sviluppata, in anticipo rispetto ad altre nazioni europee, si può notare che il linguaggio LOGO, nonostante l'onnipresenza del BASIC, sta acquistando un ruolo da protagonista, creando vaste aree applicative, del tutto nuove. Indubbiamente, se si vogliono ottenere risultati di rilievo, occorre orientarsi verso un unico linguaggio principale (e il LOGO pare essere attualmente il migliore), creando una struttura di assistenza per la diffusione nell'ambito scolastico. Se attenersi a uno standard nell'hardware usato è importante, è altrettanto indispensabile adottare un linguaggio di programmazione comune.
L'insegnamento della programmazione, in sostanza, non va preso come frutto di una moda passeggera (anche se l'impressione ricavata esaminando l'evoluzione commerciale del settore è questa): instillare anche nei più giovani i principi della programmazione significa dar loro uno strumento che incoraggia la creatività da un lato e l'abitudine a pensare in modo coerente e ben organizzato dall'altro. Infine, non va sottovalutato il ruolo complementare che l'informatica può svolgere nell'apprendimento di altre materie.

Una "chiave" da cercare

I criteri per estrarre informazioni da un archivio gestito da computer

Una chiave è una specie di strumento, incorporato nel programma per la gestione del database, che permette di individuare uno o più record specifici. Si supponga di voler consultare un manuale per la manutenzione di un'automobile alla ricerca di informazioni sul carburatore: il paragrafo corrispondente si troverà in una particolare sezione del manuale, ad esempio a pagina 36. Questo numero può essere considerato la chiave dell'informazione che stiamo cercando. Se non sappiamo che il carburatore viene descritto a pagina 36, dobbiamo consultare l'indice che ci permette di andare direttamente al paragrafo desiderato senza dover sfogliare l'intero manuale.
Il DBM segue un procedimento molto simile: ciascun record di un file di database è contrassegnato da un proprio numero (detto "chiave principale"). Quindi, per individuare un particolare record, possiamo esaminare in sequenza tutti i record del database oppure, se ne conosciamo il numero, portarci direttamente sul record desiderato. Un terzo metodo è quello di adoperare uno dei campi come "chiave secondaria" del record. Nel database relativo a un magazzino di ricambi, ad esempio, il campo usato come chiave potrebbe essere quello di COMPONENTE.
La maggior parte dei DBM consente di usare come "campi chiave" solo particolari campi del record. Quando un campo viene definito come chiave secondaria (nel nostro caso COMPONENTE), il DBM predispone una tabella delle parole (stringhe di caratteri) in esso contenute e le associa al rispettivo numero di record (chiave principale), cosicché quando viene cercato il record relativo ai carburatori, il DBM scorre la tabella COMPONENTE fino ad incontrare la stringa carburatore, legge il numero del record associato ed estrae il record richiesto. In un semplice DBM scritto in BASIC, il procedimento appena descritto può essere codificato nel modo seguente:

```
INPUT 'INDICA IL CAMPO CHIAVE';CHIA$
INPUT 'INSERISCI LA PAROLA DA CERCARE';PAR$
GOSUB 20000: REM SUBROUTINE DI RICERCA
PRINT NUMREC
```

Questa sezione di programma permette di selezionare una particolare matrice in base alla variabile CHIA$, quindi chiama una subroutine che usa la stringa PAR$ come chiave per eseguire una ricerca su tale matrice. Se l'algoritmo di ricerca è abbastanza sofisticato, la subroutine può operare anche in presenza di piccoli errori di battitura ed estrarre il record più affine. Una routine semplice come quella vista può fare a meno di una tabella a parte, con le chiavi.

I primi DBM adottati sui microcomputer erano programmi piuttosto semplici e potevano essere scritti in BASIC in un tempo ragionevolmente breve. La situazione è radicalmente cambiata con la comparsa del QL Sinclair, basato sul processore 68000. La Psion Ltd., lavorando su potenti microcomputer VAX, ha prodotto per questo apparecchio il pacchetto *Archive* che offre funzioni notevolmente avanzate per la gestione del database. Per capire meglio il funzionamento di *Archive*, creiamo un semplice database per la manutenzione di un'automobile, composto da 4 record. Ogni record contiene i due campi COMPONENTE e MANUTENZIONE:

```
Carburatore
Rimuovere la calotta e ruotare la manopola

Serbatoio dell'olio
Sollevare il coperchio e versare l'olio

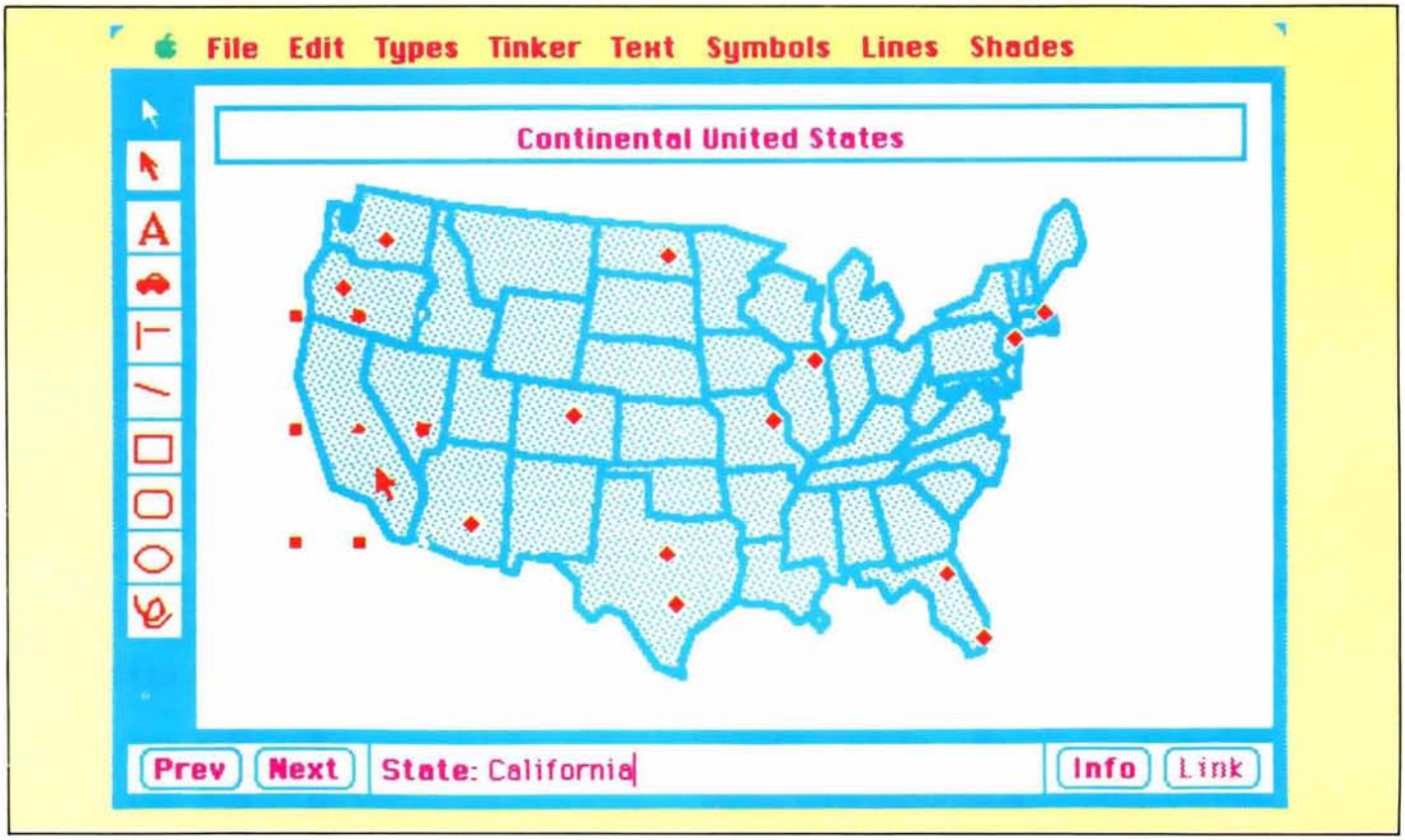
Batteria
Svitare i tappi e riempire d'acqua distillata

Tergicristalli
Svitare il tappo e riempire d'acqua
```

Quando viene lanciato, l'*Archive* suddivide lo schermo del QL in tre aree: la fascia superiore è dedicata all'immissione dei comandi, l'area centrale è quella "di lavoro", mentre il settore inferiore è riservato alla visualizzazione. Il nuovo file di database per la manutenzione dell'automobile viene creato con il comando CREATE 'AUTO'<CR> ('AUTO' è il nome assegnato al file). Quindi vengono inseriti i nomi dei campi:

```
COMPONENTE$<CR>
MANUTENZIONE$<CR>
<CR>
```

Il simbolo $ che segue i nomi di campo indica che i campi sono composti da stringhe di caratteri. Il <CR> finale conclude la definizione. Per aggiungere nuovi record al database si deve digitare il comando INSERT<CR>, che visualizza i nomi dei campi nell'area di lavoro e permette di inserire i dati relativi a ciascun campo. Premendo il tasto funzione F5 il record viene registrato nel file del database. Terminata la fase di immissione dei record, occorre premere il tasto Escape, per abbandonare il modo INSERT e impartire il comando CLOSE per uscire dal file.
La ricerca di un record contenuto in un file di *Archive* inizia con l'apertura del file tramite due comandi alternativi: LOOK consente la sola lettura dei record mentre OPEN permette di estrarre e modificare i record. Dopo aver



aperto il file, si possono usare istruzioni come FIRST, LAST, NEXT e BACK che visualizzano rispettivamente il primo record del file, l'ultimo, il record precedente e quello successivo.
Il comando di ricerca FIND deve essere seguito da un parametro o "argomento": ad esempio FIND 'CARBURATORE'. Questo comando legge tutti i campi fino a trovare una stringa identica a quella tra virgolette. L'argomento del comando può contenere anche gli operatori logici AND e OR. L'istruzione SEARCH COMPONENTE$ = 'CARBURATORE' AND MANUTENZIONE$ = 'RUOTARE' legge in successione i record del file finché non trova quello che contiene la stringa CARBURATORE nel campo COMPONENTE$ e la stringa RUOTARE nel campo MANUTENZIONE$. Se, al posto di AND, si usa l'operatore OR, il programma estrarrà il record che contiene la stringa specificata per il primo campo oppure quella specificata per il secondo.
Di solito l'inserimento dei record nel file avviene in modo casuale, ma spesso è necessario richiamarli o stamparli secondo un ordine particolare. Supponiamo di dover classificare in un database tutti i libri di una biblioteca registrandone l'autore, il titolo, l'editore e il numero di codice ISBN. Il sistema più semplice è procedere di scaffale in scaffale, registrando i volumi secondo la loro disposizione nella biblioteca. Terminato il lavoro, i record possono essere riordinati alfabeticamente (in base all'autore, al titolo, ecc.) o numericamente (in base al codice ISBN) usando le funzioni di ordinamento offerte dal DBM.
Inoltre è possibile combinare tra loro un ordinamento principale ed uno secondario. Supponiamo di voler stampare il file in ordine alfabetico per autore, classificando in base al titolo anche le diverse registrazioni relative ad ogni autore. Il risultato dovrebbe essere una lista di questo tipo:

```
ADAMI, D.M.
    "Come allevare le anatre"
    Gamma Pubblicazioni
    0-85527-435-2
ADAMI, D.M.
    "Migrazione delle anatre"
    Contacci Editore
    0-7195-1332-4
ADAMI, D.M.
    "Le anatre in cucina"
    Gamma Pubblicazioni
    0-85527-438-6
ADAMI, E.
    "Armonia nelle vocali delle lingue mediterranee"
    Edizioni Zetadelta
    52719-226-0
```

Se i nomi dei campi sono AUTORE$, TITOLO$, EDITORE$ e ISBN, è possibile riordinare i record prima per autore e poi per titolo usando questa semplice istruzione:

```
ORDER AUTORE$:A,TITOLO$:A
```

In questo esempio, la lettera A indica che la lista deve essere visualizzata o stampata in ordine alfabetico ascendente.
I DBM permettono dunque di semplificare e velocizzare lavori che altrimenti richiederebbero molto tempo, come la ricerca dei record e il loro riordinamento secondo specifici criteri. Altre sofisticate funzioni offerte da molti DBM vengono esaminate in seguito.

L'America a portata di... dita

Il "Filevision" è stato creato per i computer Apple II e III, ma le sue caratteristiche si apprezzano soprattutto nella versione per Macintosh. Il package usa il mouse dell'apparecchio per accedere alle informazioni registrate nel database, che è corredato da un programma grafico che permette di creare immagini e simboli grafici da abbinare ai campi del database. La figura in alto riproduce l'immagine di apertura per il database geografico sugli Stati Uniti. Posizionando il cursore a freccia sulla California e premendo il tasto del mouse, il programma visualizza un sommario informativo di questo stato. Premendo il tasto due volte, si apre una finestra testuale che contiene una maggiore quantità di informazioni. I dati immagazzinati possono essere combinati e riordinati come su un comune database. Selezionando l'opzione "Highlight Some" dal sottomenu "Tinker", si può specificare i criteri di ricerca (ad esempio, tutti gli stati con un numero d'abitanti superiore ai 10 milioni). In base ad essi, il programma visualizza i risultati evidenziando gli stati selezionati con linee di confine più marcate. Per stampare le informazioni basta impartire il comando PRINT prima di definire i criteri di ricerca

Muscoli e presa

Un'altra tappa nella realizzazione del nostro braccio-robot

Fase 1: Montaggio della presa

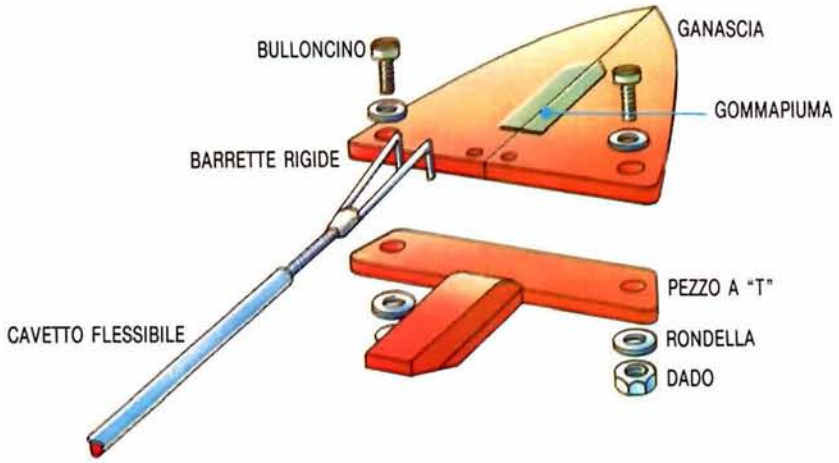
Forare le due estremità del pezzo a "T" e i pezzi della presa nei punti indicati. Incollare i cuscinetti di gommapiuma sulle ganasce della presa. Praticare due fori più piccoli alla base dei pezzi della presa: questi accoglieranno il collegamento superiore del cavetto flessibile che aziona la presa. Fissare le ganasce al pezzo a "T" con bulloncini e dadi inserendo, sopra e sotto, due rondelle in modo che le ganasce possano ruotare liberamente sui bulloncini. Saldare le due barrette rigide all'estremità superiore del cavetto flessibile (lunghezza 50 cm.). Piegare le estremità libere delle barrette ad angolo retto verso il basso e inserirle negli appositi fori nei pezzi della presa. Il collegamento composto dalle barrette e dal cavo flessibile deve avere la forma di una "Y". Tenendo chiuse le ganasce della presa, fissare la guaina plastica del cavetto flessibile al pezzo a "T" con del filo di cotone e un velo di colla.

Fase 2: Asta di collegamento del braccio superiore

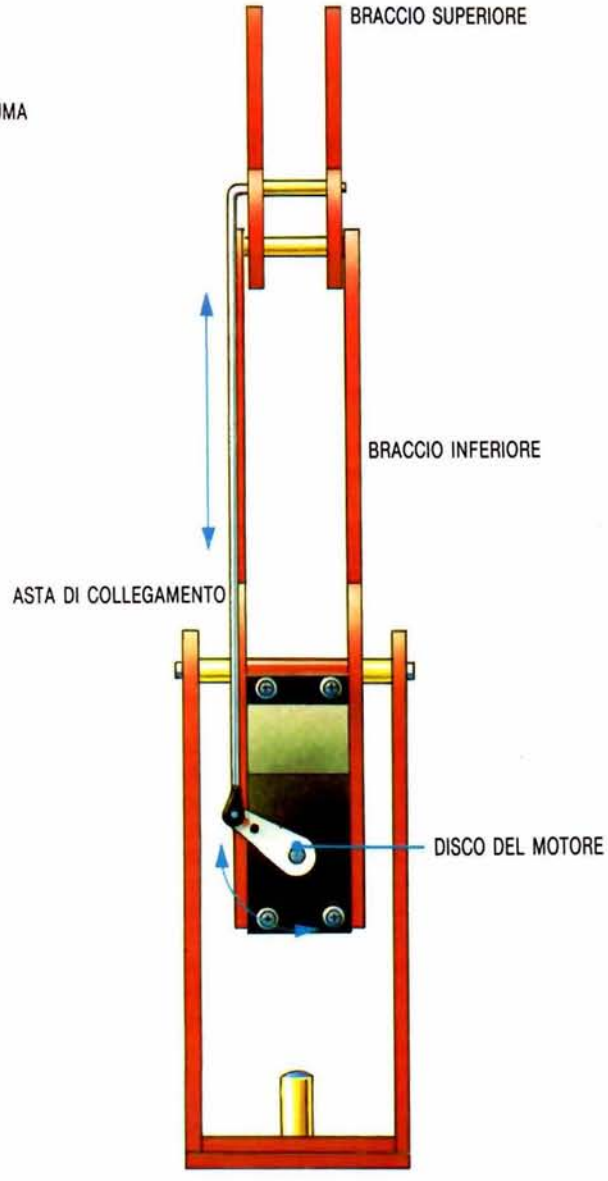
Il braccio superiore del robot è azionato dal motore montato su quello inferiore. La rotazione dell'alberino del motore viene convertita in un movimento "a bilancia" del braccio superiore, innestando sull'alberino un breve snodo o un disco di plastica del diametro di 30 mm. dotato di un foro al quale viene fissata l'asta di collegamento per il braccio superiore. Inserire il disco sull'alberino in modo che, quando il motore ruota completamente in senso anti-orario, il foro venga a trovarsi appena a sinistra delle ore 9 del quadrante di un orologio.

Prendere un'asta d'acciaio del diametro di 2 mm. e piegarne entrambe le estremità ad angolo retto; quindi inserire l'estremità inferiore nel foro del disco o snodo e quella superiore nei due fori praticati sul braccio superiore. La lunghezza dell'asta deve essere tale che i bracci inferiore e superiore formino un angolo di 90°, a livello dell'articolazione della spalla, quando il motore viene ruotato completamente in senso anti-orario. Verificare il

FASE 1: MONTAGGIO DELLA PRESA



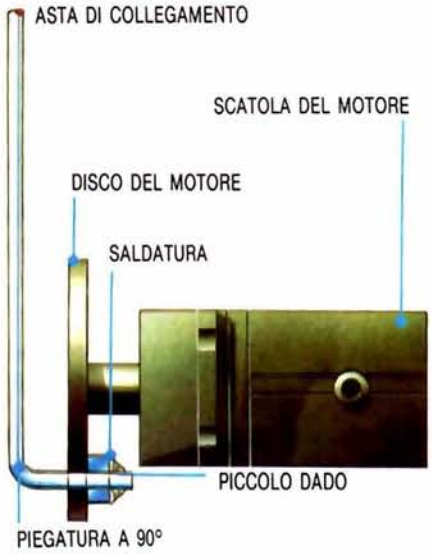
FASE 2: ASTA DI COLLEGAMENTO DEL BRACCIO SUPERIORE



COLLEGARE LE ASTE AI DISCHI

In genere chi vende servomotori dispone anche di apposite aste di collegamento. In alternativa, per fissare un'asta del diametro di 2 mm. al disco del servomotore, è necessario piegarne un'estremità a 90°, inserirla nel foro del disco e saldare un piccolo dado alla parte che sporge sul retro di questo. Tale operazione va eseguita prima di fissare il disco sull'alberino.

È consigliabile rivestire i fori praticati nel legno, che accolgono le aste di collegamento, con piccole sezioni di tubo d'ottone, di diametro interno leggermente superiore a quello delle aste. Queste guarnizioni proteggono il legno dalle abrasioni causate dal movimento delle aste



KEVIN JONES

corretto movimento dell'asta di trasmissione, facendo ruotare manualmente il disco del motore in senso orario. Quando tutto funziona a dovere, incollare o saldare l'asta al disco del motore.

Fase 3: Asta di collegamento del braccio inferiore

Montare i due motori rimasti sul corpo principale secondo lo schema della figura. Il motore di destra aziona il braccio inferiore. Ruotarlo completamente in senso antiorario e posizionare il suo disco in modo che il foro si trovi appena a destra delle ore 6 del quadrante di un orologio. Fissare le estremità di una seconda asta d'acciaio, dal diametro di 2 mm., collegandola al disco del motore e ai fori praticati sul braccio inferiore. La lunghezza dell'asta deve essere tale da posizionare orizzontalmente il braccio inferiore quando il motore viene ruotato completamente in senso anti-orario; verificare che l'asta di collegamento muova il braccio inferiore nel modo desiderato ruotando manualmente il disco del motore. Una volta controllato il corretto funzionamento, incollare o saldare l'asta al disco del motore in modo, però, da non impedirne il movimento.

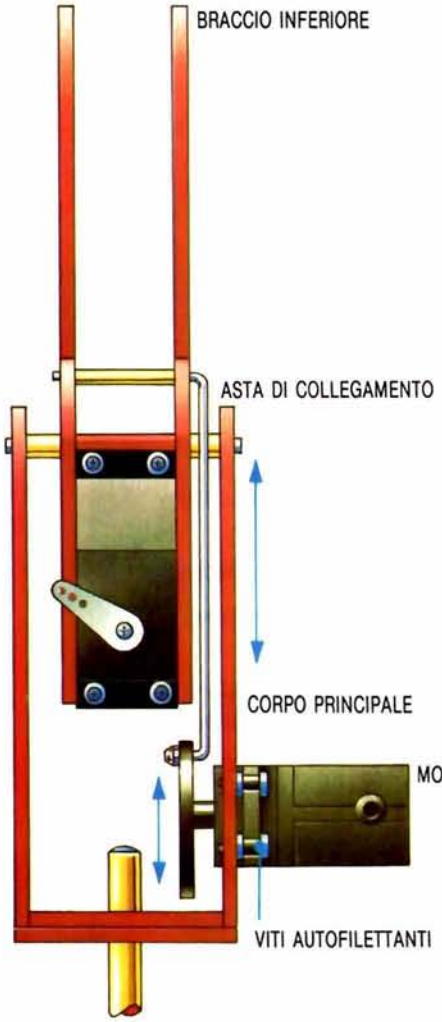
Fase 4: Collegamento del cavetto flessibile al motore

Il motore montato sulla sinistra del corpo principale apre e chiude la presa azionando il cavetto che scorre all'interno della guaina flessibile. Innanzi tutto praticare un foro nel supporto trasversale del corpo principale, inserire dal retro il cavetto flessibile e fissare la guaina plastica al foro con della colla. Chiudere saldamente le ganasce della presa tirando il cavetto interno e collegare la sua estremità al disco del motore. Quando il motore viene ruotato completamente in senso antiorario, il foro del disco deve trovarsi nella posizione delle ore 9 del quadrante di un orologio. Il sistema migliore per fissare il cavetto al disco è montare su quest'ultimo un piccolo ferma-cavi. In alternativa, piegare a 90° l'estremità del cavetto e inserirla nel foro del disco saldandola o fissandola con della colla, senza impedirne però il movimento. Indipendentemente dalla tecnica usata, il cavetto deve muoversi avanti e indietro quando il motore ruota, aprendo e chiudendo le ganasce della presa.

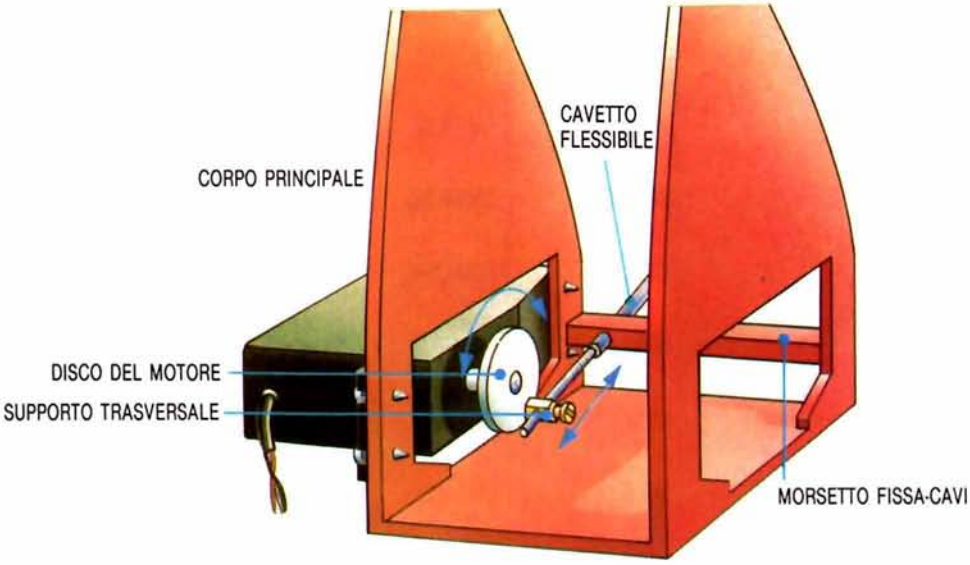
Fase 5: Asta di collegamento del corpo principale

L'ultima asta da montare collega il motore inserito nella scatola di base (vedere a pag. 1576) al corpo principale. Dopo aver sfilato il corpo principale dal perno d'acciaio, praticare un piccolo foro nella sua base. Il foro deve essere allineato al perno centrale e trovarsi in prossimità del lato destro della base (vista dall'alto). Innestare nuovamente il corpo principale nel perno e posizionarlo alle ore 2 del quadrante di un orologio (vedi figura). Ruotare completamente il motore in senso anti-orario e orientare il disco in modo che il foro si trovi subito a destra delle ore 6. Come in precedenza, fissare l'asta di collegamento e verificare che il corpo principale sia libero di girare in senso anti-orario.

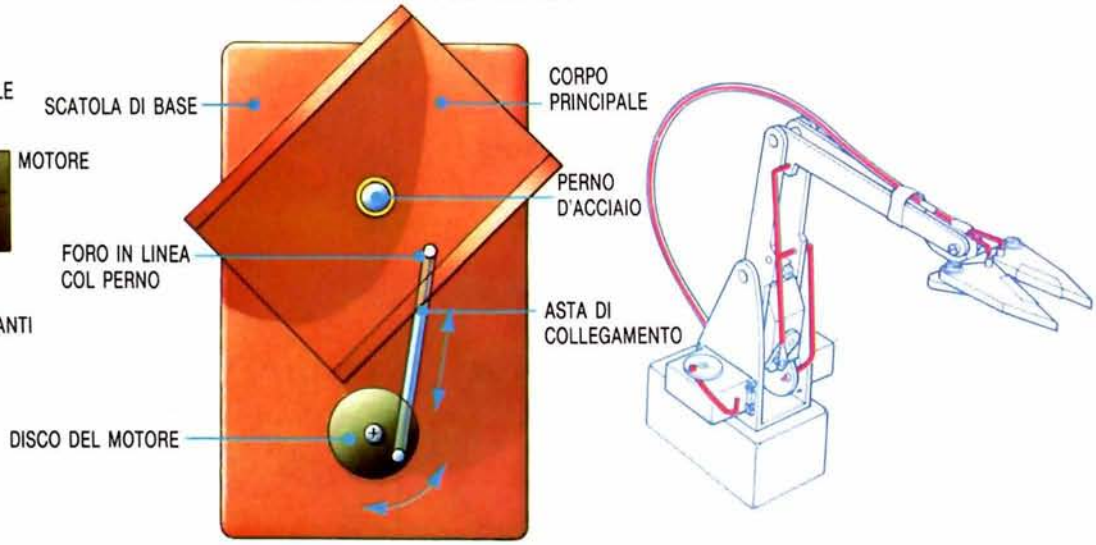
FASE 3: ASTA DI COLLEGAMENTO DEL BRACCIO INFERIORE



FASE 4: COLLEGAMENTO DEL CAVETTO FLESSIBILE AL MOTORE



FASE 5: ASTA DI COLLEGAMENTO DEL CORPO PRINCIPALE



DATABASE

La prima tabella della mappa di memoria del Commodore 64. Per concessione della Commodore Business Machines

INDIRIZZO ESADECIMALE	ETICHETTA	LOCAZIONE DECIMALE	DESCRIZIONE
0000	D6510	0	**6510 On-Chip Data-Direction Register** Registro di direzione dei dati
0001	R6510	1	**6510 On-Chip 8-Bit Input/Output Register** Registro di ingresso/uscita a 8 bit
0002		2	**Non usata**
0003-0004	ADRAY1	3-4	**Jump Vector: Convert Floating-Integer** Vettore di salto: conversione da virgola mobile a intero
0005-0006	ADRAY2	5-6	**Jump Vector: Convert Integer-Floating** Vettore di salto: conversione da intero a virgola mobile
0007	CHARAC	7	**Search Character** Ricerca di carattere
0008	ENDCHR	8	**Flag: Scan for Quote at End of String** Flag: Ricerca delle virgolette a fine di una stringa
0009	TRMPOS	9	**Screen Column From Last TAB** Colonna di schermo dopo l'ultima TAB
000A	VERCK	10	**Flag: 0 = Load, 1 = Verify** Flag: 0 = Caricamento, 1 = Verifica
000B	COUNT	11	**Input Buffer Pointer / No. of Subscripts** Puntatore al buffer d'ingresso/ N. di indici nelle matrici
000C	DIMFLG	12	**Flag: Default Array DIMension** Flag: DIMensione delle matrici in assenza di diversa specifica
000D	VALTYP	13	**Data Type: $FF = String, $00 = Numeric** Tipo di dato: $FF = Stringa, $00 Numerico
000E	INTFLG	14	**Data Type: $80 = Integer, $00 = Floating** Tipo di dato: $80 = Intero, $00 = a virgola mobile
000F	GARBFL	15	**Flag: DATA scan/LIST quote/Garbage Coll** Flag: scansione delle DATA/comando LIST/"Raccoglitore di rifiuti"
0010	000F	16	**Flag: Subscript Ref / User Function Call** Fla/Riferimento a indice/Chiamata a funzione definita dall'utente
0011	INPFLG	17	**Flag: $00 = INPUT, $40 = GET, $98 = READ** Flag: $00 = INPUT, $40 = GET, $98 = READ
0012	TANSGN	18	**Flag: TAN sign / Comparison Result** Flag: segno di TAN/Risultato di un confronto
0013		19	**Flag: INPUT Prompt** Flag: Stringa abbinata a una INPUT
0014-0015	LINNUM	20-21	**Temp: Integer Value** Provvisorio: valore intero

Il Rotronics Wafadrive

Una memoria di massa per lo Spectrum, in alternativa al Microdrive

Pur essendo uno dei più popolari home computer, lo Spectrum della Sinclair è stato frequentemente criticato per la sua scarsa utilità nelle applicazioni cosiddette "serie", limite in parte dovuto al tipo di tastiera adottata, che certamente non facilita l'impiego dei pacchetti di word processing né di gestione di database. La Sinclair Research ha così lanciato lo Spectrum+, una versione dotata di una tastiera migliore, simile a quella del QL.
Tuttavia, da un punto di vista gestionale, il nuovo modello presenta ancora molti limiti, quali la mancanza di interfacce standard e, soprattutto, di un sistema di memoria di massa veloce e affidabile. L'Interfaccia 1 e il Microdrive rispondono solo in parte a questa esigenza fondamentale: quest'ultimo, in particolare, è lento e non molto affidabile. Inoltre, mentre esiste una notevole disponibilità di software su cassetta per lo Spectrum, non si può dire lo stesso per quanto riguarda le cartucce per il Microdrive.
Tale situazione ha indotto alcune ditte indipendenti a progettare e lanciare memorie di massa alternative a quella della Sinclair. Le due unità più significative sono il Wafadrive della Rotronics, descritto in queste pagine, e il Discovery della Opus Supplies, esaminato più avanti.
Diversamente dalla combinazione Interfaccia 1/Microdrive, il Wafadrive è racchiuso in un unico contenitore che accoglie le interfacce per le periferiche e le unità di memoria di massa.
Questa struttura è più compatta, ma si dimostra meno flessibile dei Microdrive della Sinclair, che possono essere configurati "a margherita" per ampliare lo spazio di memorizzazione.

Descrizione dell'apparecchio

Il Wafadrive è racchiuso in una custodia di plastica nera dalla quale fuoriesce un cavo piatto a 35 linee, da innestare sul bus d'espansione dello Spectrum. Sul frontale dell'apparecchio si aprono due fessure per l'introduzione delle cartucce o "wafer" nella coppia di drive.
Tra le fessure si trovano tre LED: quello centrale si illumina quando l'apparecchio viene acceso mentre i LED laterali indicano l'unità in funzione.
Sul retro del Wafadrive ci sono tre connettori a pettine. Il connettore di sinistra è un bus parallelo d'espansione per il collegamento con l'Interfaccia 2; quello centrale è un'interfaccia con compatibilità Centronics per stampanti parallele mentre quello di destra è un connettore seriale RS232 per modem o altre apparecchiature seriali. Queste porte sono decisamente migliori di quelle dell'Interfaccia 1 della Sinclair che richiede, ad esempio, l'innesto di una seconda interfaccia Centronics sul connettore a pettine per potersi collegare a stampanti parallele.
Purtroppo, però, i connettori del Wafadrive (sia per le stampanti tipo Centronics sia per i modem) non sono di tipo standard, quindi occorrono cavi di collegamento realizzati appositamente.
Le cartucce o "wafer", progettate per il Wafadrive, ricordano per molti aspetti quelli per i Microdrive Sinclair. Contengono infatti un nastro magnetico ad anello continuo, largo 1,8 mm. e simile a quello delle video cassette. È stato scelto questo tipo di nastro per la sua maggiore resistenza e migliori caratteristiche di risposta, rispetto al normale nastro per musicassette. Una volta formattata, una cartuccia può contenere circa 128 Kbyte d'informazioni, ma la Rotronics produce anche wafer da 64 e da 16 Kbyte.

Memoria... alle spalle

Diversamente dal sistema di memoria di massa prodotto dalla Sinclair, il Rotronics Wafadrive è racchiuso in un'unica custodia che contiene due unità a cartucce, una porta RS232 e un'interfaccia tipo Centronics. Il collegamento al computer è assicurato da un cavo a nastro (da innestare sul connettore a pettine dello Spectrum), che permette di posizionare l'apparecchio immediatamente dietro il computer



CHRIS STEVENS

La larghezza delle cartucce è circa il doppio di quella dei wafer Sinclair, mentre la lunghezza e lo spessore dell'involucro di rivestimento sono praticamente identici. Sotto questo aspetto le cartucce Rotronics somigliano a cassette miniaturizzate. Le cartucce del Wafadrive non hanno bisogno di contenitori di protezione perché il nastro è protetto da un piccolo pannello in plastica a scorrimento automatico, simile a quello metallico dei micro-dischetti Sony da 3 1/2 pollici. Sul lato sinistro della cartuccia c'è una linguetta che può essere asportata per proteggere le informazioni sovrascritture accidentali. Una volta eliminata, però, la linguetta non può più essere riutilizzata e l'utente deve escogitare un sistema proprio che gli consenta di riabilitare la scrittura sulla cartuccia.

I comandi per il Wafadrive sono pressoché identici a quelli del Microdrive Sinclair; in entrambi i sistemi le istruzioni sono seguite dal simbolo * (ad esempio: SAVE*, LOAD*, VERIFY*, ecc.) relativo all'unità di memoria di massa esterna.

Tuttavia i comandi del Wafadrive presentano alcune differenze, poiché l'apparecchio possiede sempre e solo due drive mentre il numero di Microdrive può essere superiore. Ad esempio, per formattare una cartuccia Sinclair occorre il comando: FORMAT 'm';0;"nome", dove 'm';0 indica il numero del Microdrive in uso.

Sul Wafadrive, invece, lo stesso comando ha il seguente formato: FORMAT*'a:name', dove a: rappresenta il nome del drive in uso. Infatti il Wafadrive adopera solo gli identificatori a: e b: per specificare i due drive disponibili, mentre nel Microdrive vengono distinti da un numero compreso tra 0 e 7.

Il sistema a "flussi"

Il Wafadrive si avvale del sistema a "flussi" dello Spectrum, che riserva 16 flussi alla gestione dei circuiti di I/O, alcuni dei quali utilizzati dallo schermo e dalla stampante. I canali da 4 a 15 sono invece a disposizione delle altre periferiche. Il comando OPEN# permette di accedere ai flussi in uscita diretti al Wafadrive che, a sua volta, completa il sistema con due flussi supplementari. I canali r e c (queste lettere possono essere scritte anche in maiuscolo) sono riservati rispettivamente alle interfacce RS232 e Centronics. Il loro impiego è del tutto simile a quello dei canali t e b, usati per collegarsi alla porta RS232 sull'Interfaccia 1.

Sulla scheda del Wafadrive è montata una ROM da 8 Kbyte, contenente i comandi del BASIC "esteso" usati per controllare il sistema. Come accade per l'Interfaccia 1, il Sistema Operativo del Wafadrive (WOS) entra in funzione sostituendo gli 8 Kbyte più bassi nella ROM dello Spectrum con quelli della propria ROM.

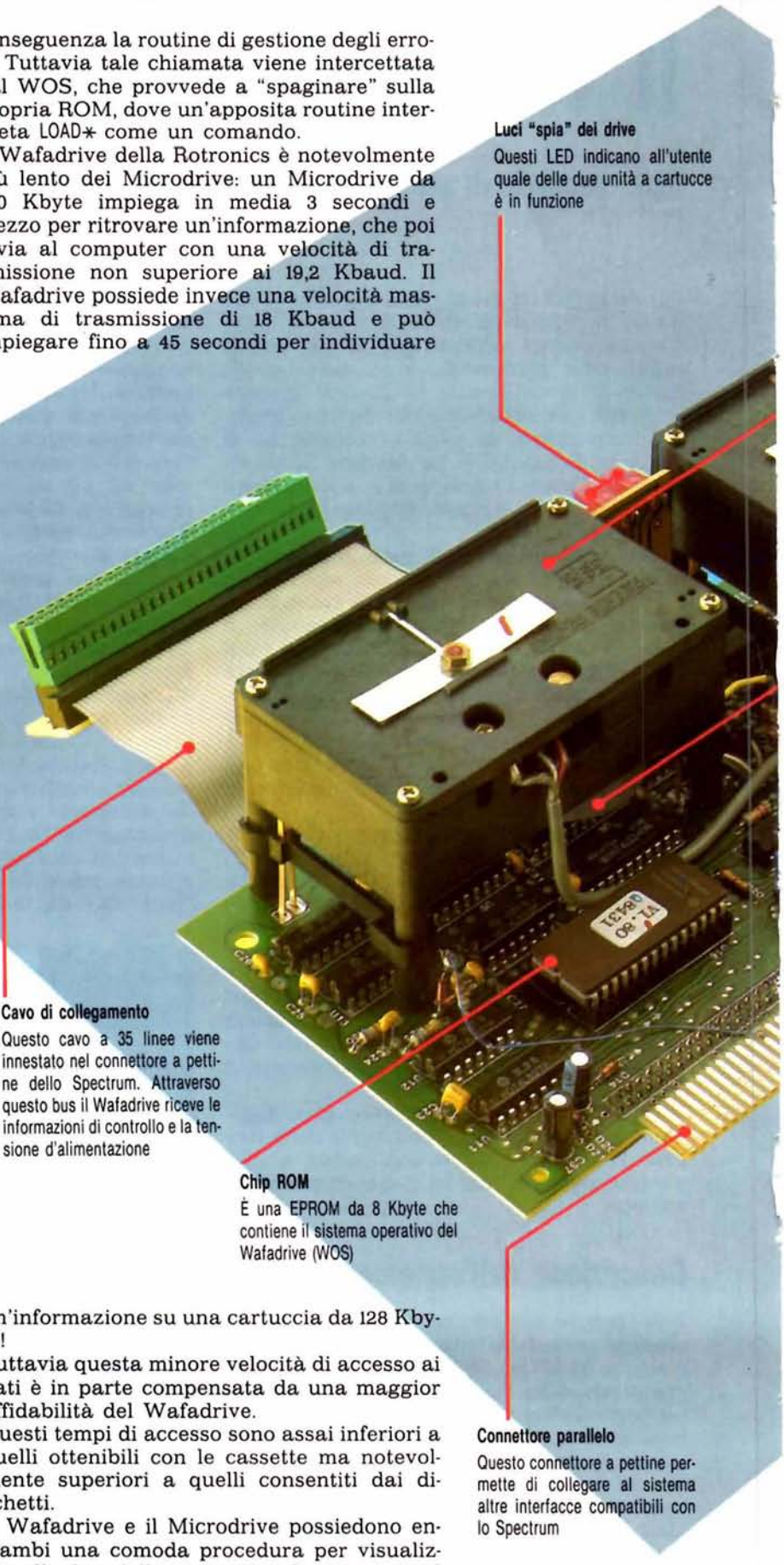
In condizioni normali, dunque, il comando LOAD* viene considerato un errore dallo Spectrum e l'interprete del BASIC chiama di conseguenza la routine di gestione degli errori. Tuttavia tale chiamata viene intercettata dal WOS, che provvede a "spaginare" sulla propria ROM, dove un'apposita routine interpreta LOAD* come un comando.

Il Wafadrive della Rotronics è notevolmente più lento dei Microdrive: un Microdrive da 100 Kbyte impiega in media 3 secondi e mezzo per ritrovare un'informazione, che poi invia al computer con una velocità di trasmissione non superiore ai 19,2 Kbaud. Il Wafadrive possiede invece una velocità massima di trasmissione di 18 Kbaud e può impiegare fino a 45 secondi per individuare un'informazione su una cartuccia da 128 Kbyte!

Tuttavia questa minore velocità di accesso ai dati è in parte compensata da una maggior affidabilità del Wafadrive.

Questi tempi di accesso sono assai inferiori a quelli ottenibili con le cassette ma notevolmente superiori a quelli consentiti dai dischetti.

Il Wafadrive e il Microdrive possiedono entrambi una comoda procedura per visualizzare l'indice della cartuccia, che si trova nel primo settore del nastro, successivo al raccordo che ne unisce le estremità. Per inviare

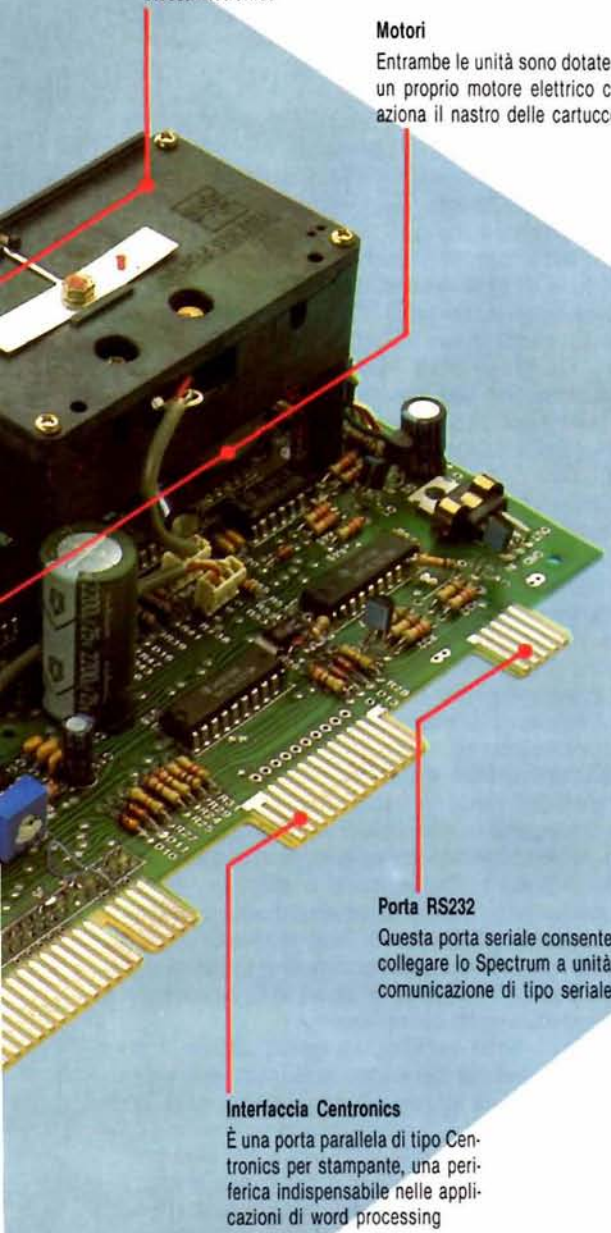


Luci "spia" dei drive
Questi LED indicano all'utente quale delle due unità a cartucce è in funzione

Cavo di collegamento
Questo cavo a 35 linee viene innestato nel connettore a pettine dello Spectrum. Attraverso questo bus il Wafadrive riceve le informazioni di controllo e la tensione d'alimentazione

Chip ROM
È una EPROM da 8 Kbyte che contiene il sistema operativo del Wafadrive (WOS)

Connettore parallelo
Questo connettore a pettine permette di collegare al sistema altre interfacce compatibili con lo Spectrum



Per incentivare l'uso di applicazioni diverse dai giochi, la Rotronics ha deciso di corredare il Wafadrive di un word processor su cartuccia, chiamato *Spectral Writer*. Questo pacchetto è abbastanza completo e permette di utilizzare al meglio le caratteristiche del Wafadrive. Le funzioni di editing (quali la formattazione di un paragrafo, l'inserimento di una parola e la cancellazione delle linee) vengono selezionate usando il tasto Symbol Shift dello Spectrum in combinazione con altri tasti.

Una serie di normali comandi consente di accedere ai file su cartuccia: le istruzioni SAVE e LOAD, ad esempio, salvano e caricano i file di testo da cassette o cartucce.

Purtroppo lo *Spectral Writer* è privo della funzione per la definizione della lunghezza della linea di schermo. Più in generale, l'efficienza del pacchetto risulta notevolmente ridotta dalla bassa qualità della tastiera dello Spectrum e dello Spectrum+.

Il successo di unità di questo genere dipende molto dalla maggiore o minore disponibilità di software. Il Wafadrive sembra essere piuttosto sfavorito perché nessuna delle più importanti software house ha sviluppato programmi su cartucce Wafadrive (ma questo problema si è presentato anche per i Microdrive Sinclair).

Esiste, comunque, un programma che consente di trasferire sui wafer Rotronics i programmi acquistati su cassetta.

Nonostante i limiti qui descritti, il Wafadrive è indubbiamente un apparecchio ben progettato e rimane la miglior alternativa al sistema Interfaccia 1/Microdrive, prodotto dalla Sinclair Research.

WAFADRIVE ROTRONICS

DIMENSIONI

mm. 230 x 110 x 80

INTERFACCE

Porta seriale RS232, interfaccia parallela Centronics, connettore "a pettine" per interfacce Sinclair

FORMATO DELLE CASSETTE

I "wafer" racchiudono un nastro magnetico ad anello chiuso

CAPACITÀ

Sono disponibili wafer da 16, 64 e 128 Kbyte

VELOCITÀ

Trasferimento dati: 16 Kbaud
Tempi di accesso massimi: 6,5 se. (cartucce da 16 Kbyte), 45 sec. (cartucce da 128 Kbyte)



CHRIS STEVENS

La cartuccia del Wafadrive

Le cartucce per il Wafadrive misurano circa il doppio di quelle per il Microdrive della Sinclair. La cartuccia o "wafer" ha le stesse proporzioni di una comune cassetta, ma il nastro al suo interno è ad anello continuo, che evita di dover riavvolgere il nastro per poter accedere ai dati già passati oltre la testina di lettura/scrittura

allo schermo l'indice di una cartuccia (comando CAT), il drive deve far girare il nastro fino a raggiungere il raccordo e leggere il settore immediatamente successivo.

Tuttavia, se il comando CAT viene ripetuto, il sistema legge l'indice direttamente dalla RAM, dove è stato registrato dopo la prima chiamata, evitando così di scorrere nuovamente tutto il nastro per tornare a leggere il primo settore. A questo punto, il WOS legge il settore successivo, per verificare che la cartuccia inserita nell'unità sia sempre la stessa; se è così, visualizza l'indice archiviato in RAM.

In alto mare

Il Capitano deve tenere il diario di bordo: ecco la routine necessaria

Durante la traversata, la velocità della nave dipende dal numero dei membri dell'equipaggio e dal loro rendimento. Ma, mentre un equipaggio numeroso e in buona salute non fa crescere la velocità della nave, quando il rendimento cala oltre un certo livello, l'inefficienza della ciurma porta ad una durata maggiore del viaggio. Se l'intero equipaggio muore, il gioco termina.
Una delle funzioni principali del Diario del Capitano è quella di comunicare al giocatore il numero dei marinai morti, l'esaurimento di una provvista e la stima aggiornata della durata del viaggio. La nuova variabile EW, creata alla linea 41, indica le settimane di viaggio supplementari, o parti di esse, che vengono sommate alle otto settimane iniziali se il rendimento dell'equipaggio cala sotto un dato livello. In questo modo la durata del viaggio può essere espressa in numeri interi, senza perdere l'eventuale parte frazionaria di EW, mentre le settimane supplementari vengono calcolate e registrate in EW come numeri reali ma sommate a JT come interi usando l'istruzione INT.
La subroutine che elabora il rapporto settimanale inizia alla linea 5300, mentre la sezione di programma relativa agli eventi che si verificano durante il viaggio viene descritta più avanti.
Poiché tali eventi possono modificare il volume delle provviste, il numero e il rendimento dei membri dell'equipaggio e la durata del viaggio, influiscono anche sui valori visualizzati da questa subroutine.
Innanzi tutto la subroutine esegue un ciclo iterativo (linea 5325) per verificare se qualche membro dell'equipaggio è morto. Esamina il secondo elemento della matrice TS(,) dei 16 membri dell'equipaggio e controlla se al loro rendimento è stato assegnato il valore −999 che indica la morte. L'eventuale decesso viene sommato a X, che aggiorna settimanalmente il numero delle vittime.
Se il valore letto sulla matrice è diverso da −999, il ciclo passa al valore successivo. Registrato il decesso del membro dell'equipaggio, la linea 5340 ne azzera il rendimento e le caratteristiche, per non rischiare di conteggiare una seconda volta questa morte nel successivo rapporto settimanale. Quindi il marinaio viene sottratto dal totale dell'equipaggio CN. Se alla fine del ciclo il valore di X è ancora 0, non si è verificato alcun decesso e il controllo passa alla linea 5400 che verifica il livello delle provviste. Se sono morti tutti, il programma visualizza un messaggio per informare che a bordo non è rimasto nessuno e che la nave va alla deriva in pieno oceano. Se invece qualche marinaio è ancora in vita, la linea 5390 visualizza il numero dei decessi.
Le provviste esaurite vengono registrate con lo stesso sistema usato per i decessi. Il programma usa un secondo ciclo, ripetuto quattro volte, per verificare il livello delle scorte. Se una di esse è terminata, il suo valore, contenuto nella matrice PA(), viene posto a −999 dalla subroutine "Distribuzione delle razioni". La linea 5415 esamina la matrice PA() e informa che una provvista è esaurita.
Il corrispondente elemento di PA() viene azzerato per evitare la ripetizione del messaggio alla fine della settimana seguente. Se le provviste esaurite sono più di una, tra i loro nomi viene inserita la congiunzione "e". Tuttavia è possibile fare un nuovo rifornimento nella sezione del programma che tratta gli eventi del viaggio; in questo caso, al corrispondente elemento di PA() viene assegnato il nuovo valore della provvista.
Per conteggiare le settimane di viaggio supplementari, il programma esamina la matrice del rendimento dell'equipaggio, calcola il rendimento totale e lo confronta con il valore soglia del rendimento ottimale. Per continuare il viaggio in condizioni di massima efficienza, il rendimento complessivo deve essere almeno 800, equivalente a 8 marinai a pieno rendimento oppure a 16 marinai a rendimento dimezzato. La linea 5465 crea un ciclo che esamina ciascun elemento della matrice dell'equipaggio mentre la linea 5470 accumula in X il rendimento complessivo.
La linea 5480 verifica se quest'ultimo è maggiore di 799. In tal caso, la durata del viaggio non cresce e il programma salta alla linea 5494, che invita il giocatore a premere un tasto per passare alla settimana successiva.
Se il rendimento totale è inferiore a 800, l'espressione alla linea 5489 calcola le settimane di viaggio supplementari. Il risultato viene registrato in EW come numero reale. La linea 5490 somma la parte intera di EW alla lunghezza del viaggio; in questo modo la durata del viaggio ancora da compiere non viene visualizzata in frazioni di settimana. Il valore contenuto in EW viene aggiunto ogni settimana alla durata del viaggio. Per evitare che la subroutine alla linea 9300 sottragga il valore WF anche al rendimento dei membri dell'equipaggio deceduti, inserire:

```
9315 IF TS(S1,2) = 0 THEN 9340
```

Come già ricordato, gli eventi che si verificano durante la traversata influiscono sul rendimento dell'equipaggio. L'unico fattore sin qui esaminato è stato il vettovagliamento. Poiché l'equipaggio riceve razioni identiche, un'ali-

MODULO 5: DIARIO DEL CAPITANO

Rapporto di fine settimana

```
5300 REM RAPPORTO SETTIMANALE
5305 PRINT CHR$(147)
5310 S$="       DIARIO DEL CAPITANO*": GOSUB 9100
5312 S$="       ---------------------*": GOSUB 9100
5314 GOSUB9200
5316 PRINT: PRINT"          FINE DELLA";WK;
     "SETTIMANA"
5318 GOSUB9200
5320 X=0
5325 FORT=1TO16
5330 IFTS(T,2)<>-999 THEN 350
5335 X=X+1
5340 TS(T,1)=0:TS(T,2)=0
5345 CN=CN=1
5350 NEXT
5355 IFX=0 THEN 5400
5358 PRINT:PRINT
5360 S$="DURANTE L'ULTIMA SETTIMANA*": GOSUB 9100
5365 IFCN>0 THEN 5390
5367 S$="SONO MORTI GLI ULTIMI*": GOSUB 9100
5368 S$="MEMBRI DELL'EQUIPAGGIO*": GOSUB 9100
5369 GOSUB9200: PRINT: GOSUB 9200
5373 S$="LA NAVE VA ALLA DERIVA*": GOSUB 9100
5374 GOSUB9200
5376 S$="IN PIENO OCEANO.....*": GOSUB 9100
5378 GOSUB9200
5380 PRINT: PRINT
5382 S$="   IL GIOCO E' TERMINATO*": GOSUB 9100
5384 PRINT: PRINT
5386 GOSUB9200: END
5388 GOTO5386
5390 PRINTX;
5392 IF X=1 THEN S$="MARINAI MORTI IN PRECEDENZA*"
5394 IF X<>1 THEN S$="MARINAI MORTI QUESTA
     SETTIMANA*"
5396 GOSUB9100
5398 GOSYB9200
5400 PRINT:PRINT
5405 S$="SONO TERMINATE LE PROVVISTE DI "
5410 FOR T=1 TO 4
5415 IF PA(T)<>-999 THEN 5440
5420 PRINT S$; P$(T)
5425 PA(T)=0
5428 S$="                          E "
5440 NEXT
5450 GOSUB 9200
5455 REM CALCOLA LA NUOVA DURATA DEL VIAGGIO (JL)
5460 W=0
5465 FOR T=1 TO 16
5470 X=X+TS(T,2)
5475 NEXT
5480 IFX>799 THEN 5494
5481 PRINT: PRINT
5482 S$="L'EQUIPAGGIO NON RENDE AL MASSIMO*":
     GOSUB 9100
5483 S$="QUINDI IL VIAGGIO DURERA' DI PIU'!!:"
GOSUB 9100
5489 EW=EW+((800-W)/800)
5490 JL=JL+INT(EW)
5492 PRINT
5494 S$=K$: GOSUB9100
5496 GET I$: IF I$="" THEN 5496
5499 RETURN
```

Variabile per le settimane supplementari

```
41 EW=0: REM SETTIMANE SUPPLEMENTARI
```

Da aggiungere al Ciclo principale del viaggio

```
880 GOSUB 5300: REM RAPPORTO DI FINE SETTIMANA
```

Basic & Basic

Spectrum:

Inserire le seguenti modifiche:

```
5305 CLS
5496 LET I$=INKEY$: IF I$="" THEN GOTO 5496
```

Micro BBC:

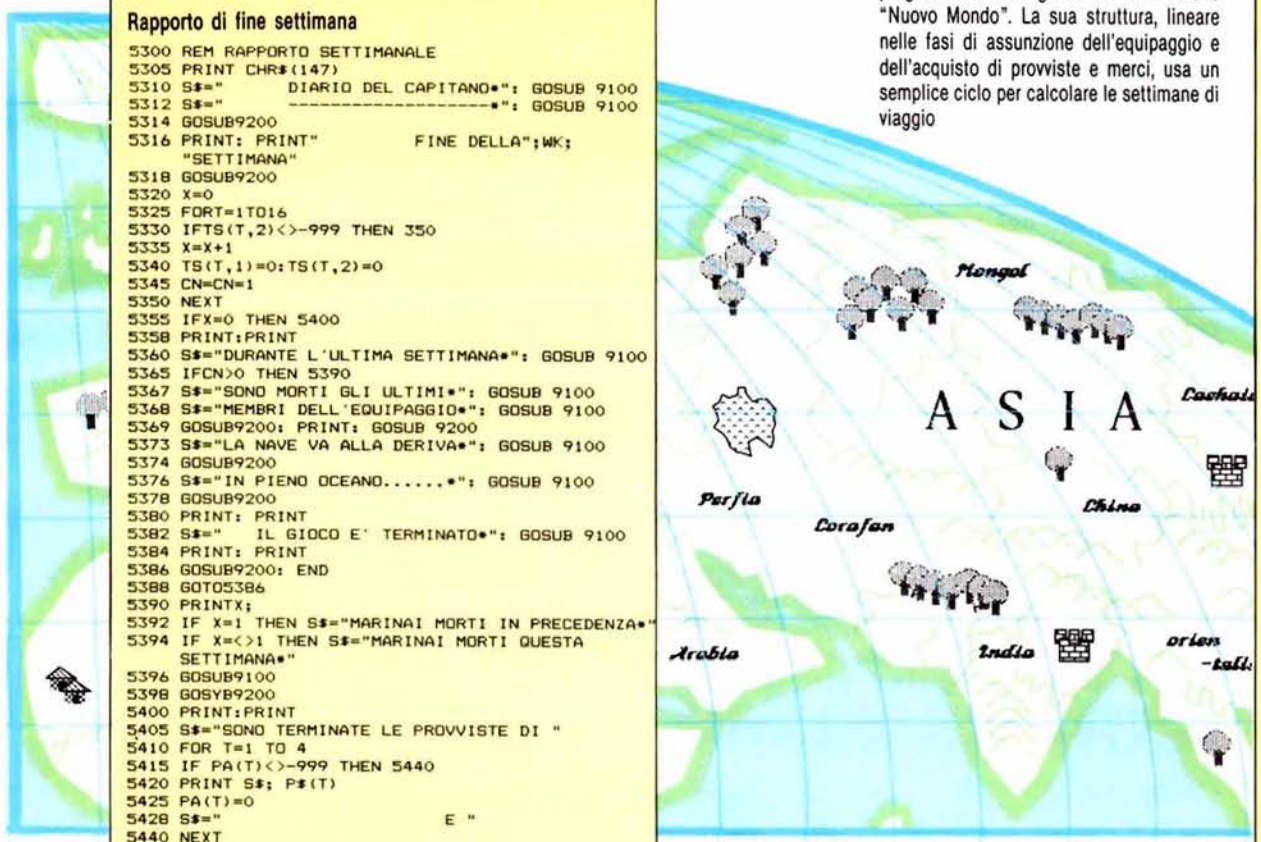
Inserire le seguenti modifiche:

```
5305 CLS
5496 I$=GET$
```

IAN McKINNELL

Riassunto... a "blocchi"

Dopo aver inserito il "Modulo 5", riassumiamo con un diagramma a blocchi le fasi già programmate del gioco di simulazione "Nuovo Mondo". La sua struttura, lineare nelle fasi di assunzione dell'equipaggio e dell'acquisto di provviste e merci, usa un semplice ciclo per calcolare le settimane di viaggio





mentazione insufficiente riduce il rendimento di tutti gli uomini. Vediamo ora quali sono le conseguenze di una riduzione individuale del rendimento in questa fase di gioco introducendo all'inizio del viaggio una routine che seleziona in modo casuale il rendimento dei marinai. Questa routine è provvisoria e va eliminata prima di inserire le successive parti del programma.

```
601 REM ROUTINE DI PROVA
602 FOR T=1 TO 16
603 IF TS(T,2)=100 THEN TS(T,2)=INT(RND(1)*100)+1
604 NEXT
```

La routine esegue per 16 volte un ciclo che modifica il secondo elemento della matrice dell'equipaggio, sostituendo al livello di rendimento 100 un valore casuale compreso tra 0 e 99, fornito dalla funzione RND. A ciascun valore generato casualmente viene sommato 1, per evitare che un eventuale valore 0 assegnato al rendimento decreti la prematura morte di un membro dell'equipaggio. La routine che visualizza il rapporto di fine settimana viene chiamata da un'istruzione GOSUB interna al ciclo principale del viaggio (linee 820-889); in origine questo veniva controllato da una struttura FOR... NEXT il cui limite superiore era indicato da JL, la variabile della durata del viaggio. Siccome, però la routine modifica il valore di JL, occorre riscrivere il ciclo sostituendo le istruzioni FOR e NEXT con:

```
820 WK=1: REM CICLO PRINCIPALE DEL VIAGGIO
889 WK=WK+1: IF WK<=JL THEN 825
```

In alto mare

Il Capitano deve tenere il diario di bordo: ecco la routine necessaria

Durante la traversata, la velocità della nave dipende dal numero dei membri dell'equipaggio e dal loro rendimento. Ma, mentre un equipaggio numeroso e in buona salute non fa crescere la velocità della nave, quando il rendimento cala oltre un certo livello, l'inefficienza della ciurma porta ad una durata maggiore del viaggio. Se l'intero equipaggio muore, il gioco termina.
Una delle funzioni principali del Diario del Capitano è quella di comunicare al giocatore il numero dei marinai morti, l'esaurimento di una provvista e la stima aggiornata della durata del viaggio. La nuova variabile EW, creata alla linea 41, indica le settimane di viaggio supplementari, o parti di esse, che vengono sommate alle otto settimane iniziali se il rendimento dell'equipaggio cala sotto un dato livello. In questo modo la durata del viaggio può essere espressa in numeri interi, senza perdere l'eventuale parte frazionaria di EW, mentre le settimane supplementari vengono calcolate e registrate in EW come numeri reali ma sommate a JT come interi usando l'istruzione INT.
La subroutine che elabora il rapporto settimanale inizia alla linea 5300, mentre la sezione di programma relativa agli eventi che si verificano durante il viaggio viene descritta più avanti.
Poiché tali eventi possono modificare il volume delle provviste, il numero e il rendimento dei membri dell'equipaggio e la durata del viaggio, influiscono anche sui valori visualizzati da questa subroutine.
Innanzi tutto la subroutine esegue un ciclo iterativo (linea 5325) per verificare se qualche membro dell'equipaggio è morto. Esamina il secondo elemento della matrice TS(,) dei 16 membri dell'equipaggio e controlla se al loro rendimento è stato assegnato il valore −999 che indica la morte. L'eventuale decesso viene sommato a X, che aggiorna settimanalmente il numero delle vittime.
Se il valore letto sulla matrice è diverso da −999, il ciclo passa al valore successivo. Registrato il decesso del membro dell'equipaggio, la linea 5340 ne azzera il rendimento e le caratteristiche, per non rischiare di conteggiare una seconda volta questa morte nel successivo rapporto settimanale. Quindi il marinaio viene sottratto dal totale dell'equipaggio CN. Se alla fine del ciclo il valore di X è ancora 0, non si è verificato alcun decesso e il controllo passa alla linea 5400 che verifica il livello delle provviste. Se sono morti tutti, il programma visualizza un messaggio per informare che a bordo non è rimasto nessuno e che la nave va alla deriva in pieno oceano. Se invece qualche marinaio è ancora in vita, la linea 5390 visualizza il numero dei decessi.
Le provviste esaurite vengono registrate con lo stesso sistema usato per i decessi. Il programma usa un secondo ciclo, ripetuto quattro volte, per verificare il livello delle scorte. Se una di esse è terminata, il suo valore, contenuto nella matrice PA(), viene posto a −999 dalla subroutine "Distribuzione delle razioni". La linea 5415 esamina la matrice PA() e informa che una provvista è esaurita.
Il corrispondente elemento di PA() viene azzerato per evitare la ripetizione del messaggio alla fine della settimana seguente. Se le provviste esaurite sono più di una, tra i loro nomi viene inserita la congiunzione "e". Tuttavia è possibile fare un nuovo rifornimento nella sezione del programma che tratta gli eventi del viaggio; in questo caso, al corrispondente elemento di PA() viene assegnato il nuovo valore della provvista.
Per conteggiare le settimane di viaggio supplementari, il programma esamina la matrice del rendimento dell'equipaggio, calcola il rendimento totale e lo confronta con il valore soglia del rendimento ottimale. Per continuare il viaggio in condizioni di massima efficienza, il rendimento complessivo deve essere almeno 800, equivalente a 8 marinai a pieno rendimento oppure a 16 marinai a rendimento dimezzato. La linea 5465 crea un ciclo che esamina ciascun elemento della matrice dell'equipaggio mentre la linea 5470 accumula in X il rendimento complessivo.
La linea 5480 verifica se quest'ultimo è maggiore di 799. In tal caso, la durata del viaggio non cresce e il programma salta alla linea 5494, che invita il giocatore a premere un tasto per passare alla settimana successiva.
Se il rendimento totale è inferiore a 800, l'espressione alla linea 5489 calcola le settimane di viaggio supplementari. Il risultato viene registrato in EW come numero reale. La linea 5490 somma la parte intera di EW alla lunghezza del viaggio; in questo modo la durata del viaggio ancora da compiere non viene visualizzata in frazioni di settimana. Il valore contenuto in EW viene aggiunto ogni settimana alla durata del viaggio. Per evitare che la subroutine alla linea 9300 sottragga il valore WF anche al rendimento dei membri dell'equipaggio deceduti, inserire:

```
9315 IF TS(S1,2) = 0 THEN 9340
```

Come già ricordato, gli eventi che si verificano durante la traversata influiscono sul rendimento dell'equipaggio. L'unico fattore sin qui esaminato è stato il vettovagliamento. Poiché l'equipaggio riceve razioni identiche, un'ali-

Come nel caso di CASE, anche questa definizione rappresenta un'eccezione alla regola che impone l'uso abbinato di BEGIN e END. Qui la END chiude la dichiarazione del RECORD e, ai fini della chiarezza, conviene commentare tale parola con il nome del record, in modo da averne ben chiara la funzione. Nell'esempio appena visto, qualsiasi variabile di tipo stanza prevede tre campi, che qui appartengono a tre diversi tipi di scalare (ma potrebbero essere tutti dello stesso tipo, sia semplici sia strutturati). Poiché non esiste alcuna restrizione sui tipi utilizzabili nei campi di un record, si può addirittura avere un campo che contiene una matrice di file composti, a loro volta, da record contenenti insiemi!
La sintassi di quanto è compreso tra le parole riservate RECORD e END è identica a quella adoperata per le dichiarazioni nella sezione VAR. Qui, però, vengono definiti identificatori di campo che vanno oltre allo "scopo" dell'identificatore del record e che sono parte integrante della struttura del record; pertanto i nomi numero, ala e occupata è come se non esistessero. Ciò significa che i nomi adoperati per i campi, essendo 'locali', possono essere dei duplicati di nomi di variabili usati all'interno del programma. Esistono due soli modi per accedervi in PASCAL: tramite la notazione "puntata" o con l'istruzione WITH.
Per selezionare un particolare campo di un record tramite la notazione "puntata" il nome del record è separato da quello del campo con un semplice punto. Ad esempio:

```
ufficio.numero
```

si riferisce unicamente al campo che contiene interi. Il contenuto di un record può essere inizializzato con sequenze di questo genere:

```
read (ufficio.numero);
'ufficio.ala := Est;'
'ufficio.occupato := gente < > [ ]'
```

e così via. Si noti che si è usato un insieme vuoto, senza mai dichiararne il tipo!

L'istruzione WITH

La notazione puntata può diventare piuttosto complessa, qualora occorra accedere a molti o a tutti i campi di un record. Esiste pertanto una forma alternativa per accedere ai campi. In breve, l'istruzione WITH comunica al computer "che si desidera lavorare su un particolare record e in particolare sui campi specificati". La sintassi dell'istruzione WITH possiede un formato identico a quello di WHILE (vedere a pag. 1572), e la sequenza di assegnazioni per l'inizializzazione potrebbe essere meglio espressa con:

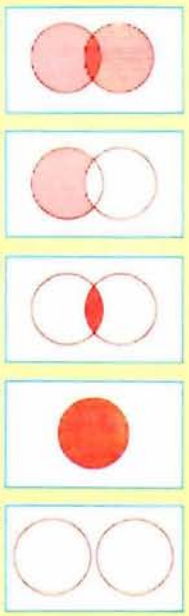
```
WITH ufficio DO
    BEGIN
        numero := 123;
        ala := Est;
        occupato := true
    END
```

Internamente al corpo dell'istruzione (tra le parole BEGIN e END in questo caso) gli identificatori di campo vengono usati senza la notazione puntata, ma utilizzandone direttamente il nome. Anche se esistessero variabili con nomi identici non può esserci confusione. Infatti lo "scopo" locale ha sempre la precedenza. Tuttavia, l'unico modo per passare valori non locali è ricorrere alla notazione puntata. Ad esempio:

OPERAZIONI SUGLI INSIEMI IN PASCAL

Ecco un elenco di tutte le possibili operazioni sui set (o insiemi), consentite dal PASCAL (a fianco è riportato, se appropriato, il relativo diagramma di Venn).

1. Unione (S1 + S2): L'insieme risultante comprende tutti gli elementi di S1 e tutti gli elementi di S2 (diagramma 1)
2. Differenza (S1 — S2): Il sottoinsieme risultante comprende i soli elementi di S1 che non appartengono anche a S2 (diagramma 2)
3. Intersezione (S1 * S2): Il sottoinsieme risultante comprende i soli elementi comuni ai due insiemi S1 e S2 (diagramma 3)
4. Equivalenza (S1 = S2): Si ha quando tutti gli elementi di S1 appartengono a S2 e viceversa (diagramma 4)
5. Diseguaglianza (S1 <> S2): Si ha quando nessun elemento di S1 appartiene a S2 e viceversa (diagramma 5)
6. Inclusione (S1 <= S2): Si ha quando tutti gli elementi di S1 appartengono anche a S2 (ma non viceversa)
7. Inclusione (S1 >= S2): È il caso opposto al precedente e si ha quando tutti gli elementi di S2 appartengono anche a S1 (ma non viceversa)
8. Appartenenza (m IN S1): Questa relazione è vera se il l'insieme a singolo elemento ([m]) è un sottoinsieme di S1



LIZ DIXON

SOLUZIONE AL PROBLEMA DEL "BREAK"

Ecco una possibile soluzione al problema presentato a pagina 1595, in cui si chiedeva di creare un programma in grado di stabilire il massimo break in una partita di biliardo

```
PROGRAM     MaxBreak   (output);

CONST
      PalleRosse = 15;

TYPE
      snooker  = (bianco, rosso, giallo, verde
                  marrone, blu, rosa, nero);
VAR
       palla : snooker;
       punti : PalleRosse .. MaxInt;
BEGIN
   punti:= PalleRosse * (ord (rosso) + ord (nero));

   FOR palla := gialla TO nera DO
       SCORE := punti + ord (palla);

   WriteLn ('Break massimo : ', punti :3)
END.
```



IAN McKINNELL

```
ufficio.numero := Numero
```

assegna il valore 'esterno' della variabile Numero all'omonimo campo del record. Si rammenti che il PASCAL non opera distinzioni tra maiuscole e minuscole. Nel nostro esempio numero o Numero si riferiscono pertanto alla variabile esterna, a meno che la parola non venga adoperata internamente a una WITH. In ogni caso, la notazione puntata è obbligatoria nell'assegnazione tra due variabili appartenenti allo stesso record, come ad esempio:

```
redazione.reparto := composizione.reparto
```

Qui l'uso di WITH diverrebbe ambiguo e pertanto non è consentito. Il miglior tentativo valido, allo stesso scopo, potrebbe essere:

```
WITH redazione DO
    reparto := composizione.reparto
```

Appare evidente, dunque, che entrambe le notazioni hanno impieghi ben precisi, condizionati dal contesto.

UNA DISTANZA COMPLESSA

Il programma "Distanze" prevede l'immissione da tastiera di due distanze, espresse in yarde e pollici, per poi visualizzare il risultato della somma. Sono state scelte queste due unità di misura per evidenziare il procedimento necessario alla somma di quantità non decimali. Per separare ciascun dato si ricorre qui alla dichiarazione di un record. La somma tiene conto dei riporti sfruttando i due operatori interi DIV e MOD. I risultati sono assegnati ai campi di Totale e poi visualizzati. L'assegnazione diretta di interi record sarebbe più rapida (ad esempio LunghA := LunghB), ma l'elaborazione concerne i singoli elementi (campi) e l'assegnazione Totale := LunghA + LunghB non sarebbe comunque consentita. Si noti l'impiego della notazione puntata e della WITH per leggere i valori di LunghA e LunghB

```
PROGRAM                 Distanze          (input, output);

CONST
        MaxByte  = 255;

TYPE
        byte     = 0 .. MaxByte;

        distanza = RECORD
                        poll    : 0 .. 11;
                        pd      : 0 .. 2;
                        yrd     : byte

                   END; {distanza}
VAR
        pollici,
        piedi    : byte;
        LunghA,
        LunghB,
        Totale   : distanza;

BEGIN

   WriteLn ('Lunghezze in yarde, piedi e pollici,');
   WriteLn ('separate da SPACE o RETURN.');
   WriteLn;
   write ('Prima : '  : 15);
   read (LunghA . yrd, LunghA . pd, LunghA . poll);
   write ('Seconda : ' : 15);

   WITH LunghB DO
     read (yrd, pd, poll);

   pollici := LunghA . poll + LunghB .poll;
   piedi := LunghA . pd + LunghB . pd + pollici DIV 12;
   WriteLn;

   WITH Totale DO
      BEGIN
         poll := pollici MOD 12;
         pd := piedi MOD 3;
         yrd := LunghA . yrd + LunghB  . yrd
                + piedi DIV 3;
         WriteLn ('Lunghezza totale:');
         WriteLn (yrd : 25, 'yrd.', pd: 1,
                  pd. e', poll : 1,' poll.')
      END
END.
```

CONTROLLO SUI DATI

Come esercizio, è stato chiesto di aggiungere al programma della Tombola (vedere a pag. 1596) una struttura iterativa per evitare la ripetizione dei numeri estratti e per individuare valori esterni alla gamma 1—90. Il miglior modo per proteggere il programma dall'immissione di numeri non validi è tramite due cicli WHILE identici, situati subito dopo le istruzioni ReadLn:

```
WHILE il numero è non valido DO
    visualizza messaggio d'errore
    ripeti la richiesta d'immissione
    ripeti l'immissione del dato
```

Nel caso dell'immissione iniziale della cartella, è necessario assicurarsi che i valori siano compresi tra 1 e 90 ed evitare duplicati. Pertanto:

```
WHILE NOT (numero IN Gamma) OR (numero IN Cartella) DO
    BEGIN
        WriteLn (numero: 20, 'non valido');
        write ('Re-immettere:':14);
        ReadLn (numero)
    END;
        {ecc.}
```

Durante l'estrazione dei numeri vale il medesimo criterio, ma la condizione per entrare nel ciclo di convalida diventa:

```
WHILE NOT (numero IN Gamma) OR (numero IN Estratto) DO
{ecc.}
```



IAN McKINNELL

Il SO del Commodore 64

Inizia una panoramica sugli aspetti utili o insoliti del sistema operativo del C64

La prima cosa che si deve conoscere di un computer, ai fini della programmazione, è la configurazione globale della sua mappa di memoria. Nel caso del Commodore 64, il microprocessore 6510, estremamente versatile, può "vedere" una tra numerose mappe possibili, a seconda del modo in cui viene configurato l'apparecchio. Infatti il 6510 può selezionare differenti banchi di memoria (vedere a pag. 628). Questa selezione può essere realizzata fisicamente, applicando una tensione di 5v o di 0v sui piedini 8 e 9 della porta d'espansione oppure via software modificando il contenuto dell'indirizzo 1. Entrambe queste operazioni modificano radicalmente il modo in cui il 6510 osserva la memoria. La figura rappresenta la normale mappa di memoria del Commodore 64 e i 64 Kbyte cui accede in condizioni normali il 6510. La tabella contiene una minuziosa descrizione della mappa di memoria.

qualora si vogliano usare i comandi BASIC PEEK e POKE, non si può certo escludere la ROM del BASIC!

Come i programmi BASIC necessitano di un'area RAM in cui registrare le proprie variabili, così le routine dell'interprete e del Kernel ricorrono alla RAM del SO, compresa tra gli indirizzi $0002 e $03FF. Essa contiene anche lo stack, situato tra le locazioni $0100 e $01FF.

La RAM del SO si trova in pagina zero poiché questa è la parte della mappa di memoria che consente un accesso più rapido alla RAM, fattore essenziale per l'attività del sistema operativo. Se si desidera abbinare un programma BASIC a uno in codice macchina, è necessario conoscere bene il funzionamento di questa area di RAM.

Esaminiamo adesso due importanti aree della RAM del SO: i puntatori del BASIC e i vettori del codice macchina.



Uno sguardo alla ROM

Poiché il processore 6510 usa indirizzi a 16 bit, può indirizzare un massimo di 65536 locazioni di memoria, ossia 64 Kbyte. Il Commodore possiede in totale 84 Kbyte, suddivisi tra ROM e RAM, ma è in grado di "vedere" solo un blocco di 64 Kbyte alla volta, sostituendo i banchi delle aree di memoria all'interno della sua mappa di memoria.

Il diagramma in figura mostra la memoria normalmente "vista" dal 6510. Per poter accedere alle aree che essa nasconde, occorre selezionare speciali registri di sostituzione. Ad esempio, se un programma è scritto completamente in codice macchina, la ROM BASIC non è necessaria e può essere eliminata dalla mappa, rendendo disponibile la RAM posta sotto di essa

Supponiamo che la memoria del Commodore 64 si trovi nello stato normale (detto "di default"). La prima cosa da osservare è che il computer possiede 64 Kbyte di RAM e 20 Kbyte di ROM. Poiché in nessun caso il 6510 può indirizzare più di 64 Kbyte degli 84 Kbyte presenti, appare evidente la necessità di una funzione che permetta di selezionare i banchi di memoria. Il diagramma della mappa di memoria mostra che la ROM dell'interprete BASIC, ossia il programma in codice macchina che esegue i programmi in BASIC, è compresa tra gli indirizzi $A000 e $CFFF mentre la ROM del Kernel, ossia il codice che gestisce tutte le funzioni di input/output, si trova tra le locazioni $E000 e $FFFF.

Sotto questi due blocchi ROM ci sono due aree RAM, ciascuna di 8 Kbyte, che in condizioni normali il 6510 non può utilizzare (sebbene istruzioni POKE o STA relative agli indirizzi di queste aree RAM vi filtrino effettivamente). Un'istruzione PEEK fornisce il contenuto della ROM: la RAM può essere letta solo escludendo dalla mappa di memoria la ROM corrispondente. Si rammenti tuttavia che,

CONFIGURAZIONE DELLA MAPPA DI MEMORIA DEL COMMODORE 64

Locazioni	Funzione
$0000-$0001	Registri di controllo della mappa di memoria del 6510
$0002-$03FF	RAM del Sistema Operativo
$0400-$07F7	RAM del video
$07F8-$07FF	Puntatori agli sprite
$0800-$9FFF	Area di programma BASIC per l'utente (variabili comprese)
$A000-$BFFF	ROM dell'interprete BASIC
$C000-$CFFF	RAM libera (4 Kbyte)
$D000-$D02E	Registri di controllo del chip VIC II (6566/9)
$D02F-$D3FF	Ripete le immagini del VIC II
$D400-$D41C	Registri di controllo del chip SID (6581)
$D41D-$D7FF	Ripete le immagini del SID
$D800-$DBE7	Nybble di colore (4 bit ciascuno)
$DBE8-$DBFF	Nybble non utilizzati
$DC00-$DC0F	Registri di controllo del chip di I/O CIA #1 (6526)
$DC10-$DCFF	Ripete le immagini del CIA #1
$DD00-$DD0F	Registri di controllo del chip di I/O CIA #2 (6526)
$DD10-$DDFF	Ripete le immagini del CIA #2
$E000-$FFFF	ROM Kernel

Puntatori del BASIC

In alcune circostanze può essere utile modificare il contenuto dei puntatori del BASIC. Per esempio, gli indirizzi $002B (43 in decimale) e $002C (44 in decimale) abitualmente contengono i valori 1 e 8. Questo è l'indirizzo iniziale del BASIC (nel formato lo-hi, o "basso-alto") e significa che il BASIC inizia dalla locazione 8x256+1=2049, ossia $0801. In realtà il BASIC comincia alla locazione $0800 ma il sistema operativo esige che il primo byte sia sempre 0 quindi il programma BASIC inizia effettivamente da $0801 (detto per inciso, $0801 è anche l'indirizzo da cui parte il salvataggio di un programma BASIC quando si usa il programma monitor per il codice macchina del Commodore).
Prima di caricare un programma BASIC è possibile modificare l'indirizzo iniziale del BASIC, intervenendo su questi due puntatori.

TABELLA DEI PUNTATORI AL BASIC DEL CBM 64

Locazioni	Funzione
$002B-$002C	Inizio area BASIC per l'utente
$002D-$002E	Inizio area variabili BASIC
$002F-$0030	Inizio area matrici BASIC
$0031-$0032	Fine area matrici BASIC + 1
$0033-$0034	Fondo dell'area stringhe
$0035-$0036	Utility
$0037-$0038	Fine area BASIC per l'utente

Occorre però avere l'accortezza di far iniziare il BASIC ad un margine di pagina (una "pagina" rappresenta un blocco di memoria di 256 byte), cioè che la locazione $002B contenga il valore 1, e che il primo byte sia uno 0. Quindi la linea:

```
POKE 2560,0: POKE 44,10: NEW
```

eseguita in modo diretto, innalza di due pagine il fondo della memoria portandolo all'indirizzo $2560; l'istruzione NEW permette di azzerare con grande rapidità tutti i puntatori (contenuti nelle locazioni da $002D a $0038). L'innalzamento del fondo del BASIC permette di avere contemporaneamente in memoria due programmi BASIC: la procedura prevede il caricamento del primo programma, l'innalzamento del fondo del BASIC e, infine, il caricamento del secondo programma.
Il seguente intervento sui puntatori, usato con maggior frequenza del precedente, permette di abbassare la sommità della memoria per fare spazio ad un programma in codice macchina. La linea:

```
POKE 56,159: POKE 51,0: POKE 52,159
```

abbassa la sommità della memoria di una pagina. Dopo aver rimosso dall'area di memoria del BASIC un blocco di RAM, si può star sicuri che il SO non userà quest'area per registrarvi le variabili BASIC: questa precauzione pone al sicuro il codice macchina da danneggiamenti accidentali.

I vettori RAM

Un secondo blocco della RAM, di particolare interesse per i programmatori in codice macchina, contiene i vettori RAM ed è compreso tra le locazioni $0314 e $0333. Un vettore RAM è paragonabile a un comune scambio ferroviario; di solito, quando un treno (o, per analogia, il processore 6510 che esegue il proprio programma) attraversa uno scambio, lo supera senza cambiare direzione di marcia. Tuttavia, in alcuni casi, può essere necessario deviarlo su una linea laterale, facendogli attraversare una o due stazioni secondarie, prima di riportarlo su quella principale.

TABELLA DEI VETTORI DEL CBM 64

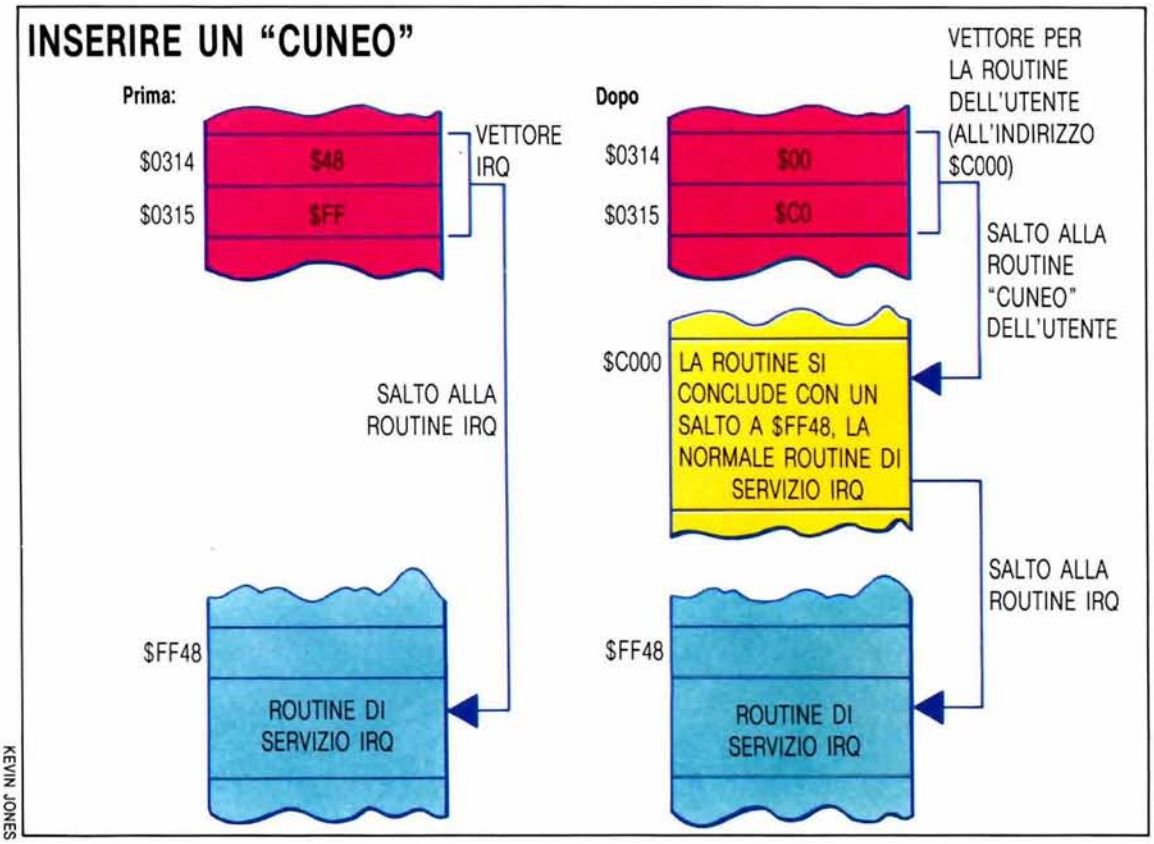
Locazioni	Vettore
$0314-$0315	IRQ (Interrupt)
$0316-$0317	BRK (Interrupt)
$0318-$0319	NMI (Interrupt)
$031A-$031B	Routine OPEN del Kernel
$031C-$031D	Routine CLOSE del Kernel
$031E-$031F	Routine CHKIN del Kernel
$0320-$0321	Routine CHKOUT del Kernel
$0322-$0323	Routine CLRCHN del Kernel
$0324-$0325	Routine CHRIN del Kernel
$0326-$0327	Routine CHROUT del Kernel
$0328-$0329	Routine STOP del Kernel
$032A-$032B	Routine GETIN del Kernel
$032C-$032D	Routine CLALL del Kernel
$032E-$032F	Definito dall'utente
$0330-$0331	Routine LOAD del Kernel
$0332-$0333	Routine SAVE del Kernel

Prendiamo ad esempio il vettore IRQ (Interrupt ReQuest). Quando il Commodore 64 funziona normalmente, ogni sessantesimo di secondo viene avviato uno dei temporizzatori del chip di I/O del 6526, che pone a livello basso la linea IRQ del 6510. Terminata l'elaborazione dell'istruzione corrente, il 6510 risponde all'"abbassamento" della linea IRQ generando una interrupt ed eseguendo una routine di servizio, che inizia all'indirizzo $FF48. Questa routine per le IRQ esamina anche la tastiera per individuare eventuali pressioni di tasti da parte dell'utente. Una delle prime istruzioni eseguite è:

```
JMP ($0314)
```

che provoca un salto indiretto all'indirizzo contenuto nelle locazioni $0314 (byte-basso) e $0315 (byte alto), che si trovano in RAM. Quindi, modificando il loro contenuto, è possibile puntare a una diversa sezione del programma. Elaborata questa sezione, il microprocessore può essere inviato all'indirizzo registrato in origine nelle due locazioni. In questo modo al 6510 viene fatta eseguire, ogni sessantesimo di secondo, una particolare sezione di codice scritta dall'utente (purché questa non sia troppo lunga, nel qual caso impedirebbe l'impiego di qualsiasi routine Kernel).

Una routine così concepita viene chiamata "cuneo", per il modo in cui si "incunea" nel normale funzionamento del sistema.



"Cunei" di tempo

Il processore 6510 viene interrotto ogni sessantesimo di secondo per eseguire operazioni di servizio quali la scansione della tastiera. L'indirizzo iniziale della routine di servizio si trova alle locazioni $0314 e $0315.
Modificando l'indirizzo contenuto in questa coppia di locazioni, è possibile "incuneare" una breve sezione di programma scritta dall'utente che viene eseguita prima della routine di servizio. Di solito, infatti, il controllo viene ripassato alla normale routine di servizio al termine della routine detta "cuneo". Pertanto il sistema esegue la routine dell'utente ogni qualvolta venga generata una interrupt, ossia ogni sessantesimo di secondo

L'OROLOGIO DEL COMMODORE

Questo programma "caricatore", in BASIC, e il successivo listato assembly illustrano come sia possibile inserire una sezione di programma "cuneo" in una normale routine di gestione delle interrupt. Il "cuneo" in questione visualizza un orologio in un angolo dello schermo. Poiché l'ora viene aggiornata ricorrendo alla normale interrupt generata ogni sessantesimo di secondo dal sistema, quest'ultimo funziona regolarmente anche durante la visualizzazione dell'orologio, permettendo di inserire ed eseguire, contemporaneamente, un programma BASIC. Per prima cosa la routine "rimette l'ora" con i tempi iniziali che le vengono comunicati, poi modifica il contenuto di $0314 e $0315 per puntare alla routine "cuneo". Questa aggiorna e visualizza l'orologio ogni volta che viene generata un'interrupt IRQ. Come operazione finale esegue un salto (JMP) alla normale routine di sistema IRQ, il cui indirizzo viene conservato nelle locazioni VETT e VETT+1.
Come alternativa all'introduzione e all'assemblaggio di questo programma "sorgente", è possibile inserire il programma sotto forma d'istruzioni DATA che contengono il codice macchina. L'esecuzione del programma "caricatore" in BASIC produce la comparsa dell'orologio nell'angolo superiore destro dello schermo. Il "checksum" alla linea 1420 visualizza un errore di checksum qualora uno dei valori nelle DATA risultasse trascritto con imprecisione.

L'orario iniziale viene selezionato con una serie d'istruzioni POKE, introducendo le ore, i minuti e i secondi, rispettivamente, nelle locazioni 50129, 50130 e 50131. Alla fine del programma di caricamento, le linee da 1480 in poi avviano l'orologio a partire dall'ora pomeridiana 4:30:00. Per modificare questo orario basta modificare i valori registrati nelle tre locazioni. SYS 50138 lancia il programma di codice macchina e avvia l'orologio, che può essere spento premendo contemporaneamente i tasti RUN/STOP e RESTORE o impartendo il comando SYS 50237 che lancia un'apposita routine acclusa al programma in codice macchina

Programma di caricamento BASIC

```
1000 REM ** CARICATORE BASIC
     DELL'OROLOGIO **
1010 DATA 173,166,2,240,10,169,128,13,14
1020 DATA 221,141,14,221,48,8,169,127,45
1030 DATA 14,221,141,14,221,169,127,45,
1040 DATA 15,221,141,15,221,173,208,195
1050 DATA 41,128,141,208,195,173,209,195
1060 DATA 32,28,197,13,208,195,141,11
1070 DATA 221,173,210,195,32,28,197,141
1080 DATA 10,221,173,211,195,32,28,197
1090 DATA 141,9,221,169,0,141,8,221,120
1100 DATA 173,20,3,141,214,195,173,21,3
1110 DATA 141,215,195,169,76,141,20,3
1120 DATA 169,196,141,21,3,88,96,120,173
1130 DATA 214,195,141,20,3,173,215,195
1140 DATA 141,21,3,88,96,193,216,195,201
1150 DATA 6,240,3,76,8,197,169,255,141
1160 DATA 216,195,173,213,195,240,243
1170 DATA 193,11,221,190,41,128,208,5
1180 DATA 169,1,76,111,196,169,16,141,68
1190 DATA 4,173,212,195,141,38,216,169
1200 DATA 13,141,39,4,173,212,195,141,39
1210 DATA 216,138,41,16,32,14,197,141,28
1220 DATA 4,173,212,195,141,28,216,138
1230 DATA 32,22,197,141,29,4,193,212,195
1240 DATA 141,29,216,169,58,141,30,4,173
1250 DATA 212,195,141,30,216,173,10,221
1260 DATA 170,32,14,197,141,31,4,193,212
1270 DATA 195,141,31,216,138,32,22,197
1280 DATA 141,32,4,173,212,195,141,32
1290 DATA 216,169,47,141,33,4,193,212
1300 DATA 195,141,33,216,173,9,221,170
1310 DATA 32,14,197,141,34,4,193,212,195
1320 DATA 141,34,216,138,32,22,197,141
1330 DATA 35,4,173,212,195,141,35,216
1340 DATA 169,46,141,36,4,173,212,495
1350 DATA 141,36,216,173,8,221,105,48
1360 DATA 141,37,4,193,212,195,141,37
1370 DATA 216,238,216,195,108,214,195,74
1380 DATA 74,74,74,24,105,48,96,41,15,24
1390 DATA 105,48,96,160,255,56,200,233
1400 DATA 10,476,251,105,10,141,217,195
1410 DATA 152,10,10,10,10,13,217,195,96
1420 DATA 42131: REM *CHECKSUM*
1430 CC=0
1440 FOR I=50138 TO 50481
1450 READ X:CC=CC+X:POKE I,X
1460 NEXT
1470 READ X:IF CC<>X THEN PRINT
     "CHECKSUM ERROR"
1480 REM ** COLLAUDO DELL'OROLOGIO **
1490 POKE 50128,128: REM AM/PM
1500 POKE 50132,8: REM COLORE
1510 POKE 50133,1: REM SCHERMO
1520 POKE 50129,4: REM ORE
1530 POKE 50130,30: REM MINUTI
1540 POKE 50131,0: REM SECONDI
1550 SYS 50138: REM CHIAMA LA ROUTINE

USARE SYS 50237, NEL MODO DIRETTO O
NEL PROGRAMMA, PER METTERE O
ELIMINARE LA ROUTINE DI "CUNEO"
```

Listato Assembly

```
;**************************************
;*                                    *
;*      PROGRAMMA "CUNEO" PER         *
;*          OROLOGIO IRQ              *
;*                                    *
;*  50128 = AM=0 / PM=129             *
;*  50129 = ORE                       *
;*  50130 = MINUTI                    *
;*  50131 = SECONDI                   *
;*  50132 = COLORE OROLOGIO           *
;*  50133 = DISPLAY ON=1/OFF=0        *
;*                                    *
;*  INSERIMENTO DEL CUNEO SYS 50138   *
;*  RIMOZIONE DEL CUNEO   SYS 50237   *
;*                                    *
;**************************************

IRQVET = $0314      ; VETTORE RAM DI IRQ
CLOCK = $DD08       ; REGISTRO TOD (ORA DEL GIORNO)
D2CRA = $DD0E       ; VIA#2 CRA
D2CRB = $DD0F       ; VIA#2 CRB
PALNTS = $02A6      ; FLAG PAL/NTSC
RITMO = $06         ; UN DISPLAY OGNI 6 IRQ
CIFRA = $30         ; CODICE SCHERMO PER '0'
PUNTO = $2E         ; CODICE SCHERMO PER '.'
BARRA = $2F         ; CODICE SCHERMO PER '/'
DUEPT = $3A         ; CODICE SCHERMO PER ':'
HZ50 = $80          ; MASCHERA A 50 HZ PER TODIN
HZ60 = $7F          ; MASCHERA A 60 HZ PER TODIN
AY = $01            ; CODICE SCHERMO PER 'A'
PEE = $10           ; CODICE SCHERMO PER 'P'
EM = $0D            ; CODICE SCHERMO PER 'M'
SCRIVI = 127        ; MASCHERA PER RIMETTERE IL CLOCK IN CR
SCNLOC = $041C      ; INDIRIZZO DI SCHERMO DELL'OROLOGIO
COLLOC = $D81C      ; INDIRIZZO IN MATRICE DEL VIDEO
TRNCLO = $0F        ; MASCHERA PER NYBBLE INFERIORE

* = $C3D0           ; INDIRIZZO INIZIALE DEL CODICE
AMPM *=*+1          ; FLAGE PER AM/PM
ORE *=*+1           ; VALORE ORE (PER INIZIALIZZAZIONE)
MINS *=*+1          ; VALORE MINUTI
SECS *=*+1          ; VALORE SECONDI
COLOR *=*+1         ; COLORE OROLOGIO
DISPLY *=*+1        ; FLAG VISUALIZZA/NASCONDI
VETTOR *=*+2        ; DEPOSITO PER VETTORE IRQ ORIGINALE
CONTO *=*           ; CONTATORE DI IRQ
TEMP1 *=*+1

;
;INSERIMENTO DEL "CUNEO"
;
    LDA PALNTS      ; PAL OPPURE NTSC
    BEQ NTSC        ; DIRAMAZIONE PER NTSC
    LDA #HZ50       ; DEVE ESSERE PAL
    ORA D2CRA
    STA D2CRA       ; FISSA TOSIN PER 50 HZ
    BMI PALDUN
NTSC
    LDA #HZ60       ; NTSC
    AND D2CRA
    STA D2CRA       ; FISSA TODIN PER 60 HZ
PALDUN
    LDA #SCRIVI
    AND D2CRB       ; SELEZIONA OROLOGIO SENZA SVEGLIA
    STA D2CRB
    LDA AMPM
    AND #128        ; RENDE IL VALORE DI AMPM VALIDO
    STA AMPM
    LDA ORE         ; ESTRARE LE ORE
    JRS BINBCD      ; CONVERTE IN BCD
    ORA AMPM        ; ESEGUE OR CON FLAG DI AM/PM
    STA CLOCK+3     ; DEPOSITA IN CLOCK
    LDA  MINS       ; ESTRAE I MINUTI
    JSR BINBCD      ; CONVERTE IN BCD
    STA CLOCK+2     ; DEPOSITA IN CLOCK
    LDA SECS        ; ESTRAE I SECONDI
    JSR BINBCD      ; CONVERTE IN BCD
    STA CLOCK+1     ; DEPOSITA IN CLOCK
    LDA #00         ; AZZERA SEMPRE I 10MI
    STA CLOCK       ; INIZIA CONTEGGIO
;
    SEI             ; DISABILITA LE INTERRUPT
    LDA IRQVET
    STA VETTOR      ; SALVA VETTORE IRQ ORIGINALE
    LDA IRQVET+1
    STA VETTOR+1
;
    LDA #,CUNEO
    STA IRQVET
    LDA #,CUNEO     ; INSERISCE IL CUNEO
    STA IRQVET+1
    CLI             ; ABILITA LE INTERRUPT
    RTS

;
;RIMOZIONE DEL CUNEO
;
    SEI             ; DISABILITA LE INTERRUPT
    LDA VETTORQ
    STA IRQVET      ; RIPRISTINA IL VETTORE IN RAM
    LDA VETTOR+1
    STA IRQVET+1
    CLI             ; ABILITA LE INTERRUPT
    RTS

;
;IL CUNEO INIZIA DA QUI
;
CUNEO
    LDA CONTO       ;
    CMP #RITMO      ; ESEGUE CLOCK A QUESTA IRQ?
    BEQ CONT        ;
ESCI
    JMP FINE        ; NO
CONT
    LDA #$FF        ; RESETTA CONTATORE DI IRQ
    STA CONTO
    LDA DISPLY      ; DISPLAY?
    BEQ ESCI        ; NO...DIRAMA

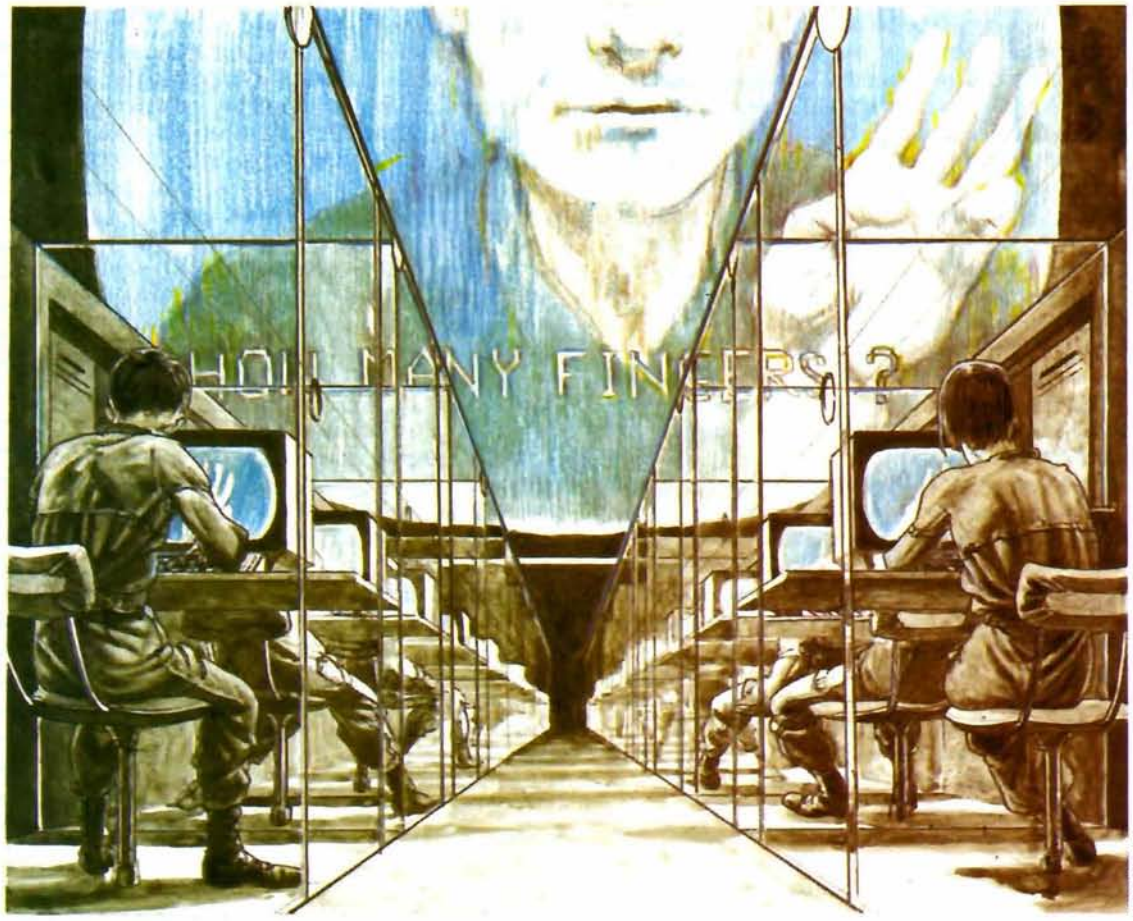
    LDA CLOCK+3     ; ESTRAE ORE/AM/PM
    TAX             ; COPIA NEL REGISTRO X
    AND #$80        ; ESTRAE AM/PM
    BNE PM          ; DIRAMAZIONE SE PM
    LDA #AY         ; VISUALIZZA 'A'
    JMP MERIDP
PM
    LDA #PEE        ; VISUALIZZA 'P'
    MERIDP
    STA SCNLOC+10
    LDA COLOR       ; ESTRAE COLORE
    STA COLLOC+10   ; SELEZIONA IL COLORE
    LDA #EM         ; VISUALIZZA 'M'
    STA SCNLOC+11
    LDA COLOR       ; SELEZIONA IL COLORE
    STA COLLOC+11
;
;CICLO PER LE ORE
;
    TXA             ; ESTRAE LE ORE
    AND #$10        ; SOLO LA CIFRA ALTA
    RSR HICIFRA     ; CONVERTE IN CODICE PER LO SCHERMO
    STA SCNLOC      ; LO VISUALIZZA
    LDA COLOR
    STA COLLOC      ; SELEZIONA IL COLORE
    TXA             ; RIPRENDE IL BYTE
    JRS LODIGT      ; CIFRA PIU' BASSA
    STA SCNLOC+1    ; LA VISUALIZZA
    LDA COLOR
    STA COLLOC+1    ; SELEZIONA IL COLORE
;
    LDA #DUEPT      ; SEPARAZIONE ORE/MINUTI
    STA SCNLOC+2:
    LDA COLOR
    STA COLLOC+2
;
;ADESSO ESEGUE IL CICLO PER I MINUTI
;
    LDA CLOCK+2     ; ESTRAE I MINUTI
    TAX
    JSR HIDIGT      ; CIFRA ALTA
    STA SCNLOC+3    ; LA VISUALIZZA
    LDA COLOR
    STA COLLOC+3    ; E SELEZIONA IL COLORE
    TXA
    JSR LODIGT      ; CIFRA BASSA
    STA SCNLOC+4    ; LA VISUALIZZA
    LDA COLOR
    STA COLLOC+4    ; SELEZIONA IL COLORE
;
    LDA #BARRA      ; SEPARATORE MIN/SEC
    STA SCNLOC+5
    LDA COLOR
    STA COLLOC+5
;
;ADESSO ESEGUE IL CICLO PER I SECONDI
;
    LDA CLOCK+1     ; ESTRAE I SECONDI
    TAX
    JSR HIDIGT      ; CIFRA ALTA
    STA SCNLOC+6    ; LA VISUALIZZA
    LDA COLOR
    STA COLLOC+6    ; E SELEZIONA IL COLORE
    TXA             ; RIPRENDE IL BYTE
    JSR LODIGT      ; CIFRA BASSA
    STA SCNLOC+7    ; LA VISUALIZZA
    LDA COLOR
    STA COLLOC+7    ; E SELEZIONA IL COLORE
;
    LDA #PUNTO      ; SEPARATORE SECONDI/DECIMI
    STA SCNLOC+8
    LDA COLOR
    STA COLLOC+8
;
;ADESSO ESEGUE CICLO PER I DECIMI
;
    LDA CLOCK       ; ESTRAE DECIMI DI SECONDO
    ADC #CIFRA      ; SOMMA $30 PER CODICE SCHERMO
    STA SCNLOC+9    ; VISUALIZZA
    LDA COLOR
    STA COLLOC+9    ; E SELEZIONA IL COLORE
;
;
FINE
    INC CONTO       ; INCREMENTA CONTATORE IRQ
    JMP (VETTOR)    ; ESEGUE NORMALE ROUTINE DI IRQ

;
;SUBROUTINE
;
HICIFRA
    LSR A
    LSR A           ; SPOSTA NYBBLE ALTO NEL BASSO
    LSR A
    LSR
    CLC
    ADC #CIFRA      ; SOMMA #30 PER CODICE SCHERMO
    RTS

LODIGT
    AND #TRNCLO     ; RIMUOVE IL NYBBLE ALTO
    CLC
    ADC #CIFRA      ; SOMMA #30 PER CODICE SCHERMO
    RTS

;
;CONVERSIONE DA BINARIO A BCD
;
BINBCD
    LDY #$FF
    SEC
D10
    INY
    SBC #10         ; SOTTRAE FINO A VALORE NEGATIVO
    BCS D10
    ADC #10         ; SOMMA 10 AL RISULTATO
    STA TEMP1       ; DEPOSITA IL RESTO
    TYA             ; NUMERO DI 10 SOTTRATTI
    ASL A
    ASL A
    ASL A           ; SCORRIMENTO NEL NYBBLE ALTO
    ASL A
    ORA TEMP1       ; METTE IL RESTO NEL NYBBLE BASSO
    RTS
```

Riprodotto per gentile concessione degli Autori e della Ellis Horwood Ltd. Tratto da: Mastering the Commodore 64 di Jones & Carpenter



NICK HARRIS

Indottrinazione interattiva

La diffidenza verso il CAL, nell'insegnamento tramite computer, è dovuta più allo "spauracchio" creato dai celebri scritti orwelliani e dai poco felici esperimenti iniziali, che non dalla realtà attuale. I computer, i collegamenti a rete e i videodischi costituiscono validi strumenti, che formeranno un'efficace struttura dell'insegnamento per gli anni a venire

Il Computer Assisted Learning

Questa definizione viene oggi applicata a qualsiasi insegnamento tramite computer, ma...

L'insegnamento tramite computer (o più brevemente CAL) è noto anche con le sigle CAI (Computer Aided Instruction) e CBL (Computer Based Learning). In pratica tutte queste sigle significano la stessa cosa: istruzione effettuata con l'aiuto di computer e relativo software.

Qui di seguito tratteremo in particolare del CAL riservato ai programmi di insegnamento e non di quello riservato ai giochi, alle simulazioni o ad altri programmi per l'infanzia. Nel CAL il computer è adoperato per trasmettere informazioni, per verificare se queste sono state apprese e per regolare gli esercizi pratici. In altre parole è usato come "macchina programmata per insegnare".

È opinione diffusa, anche se errata, che il CAL serva a poco o a nulla: tale concetto deriva in gran parte dai primi esperimenti di "istruzione programmata", condotti da Skinner e dai suoi successori (vedere a pag 1481). Molti pregiudizi, diffusi tra gli insegnanti, sono stati poi rafforzati dalla successiva produzione di software per CAL di qualità scadente e nella quale lo studente veniva collocato in una posizione passiva, atteggiamento questo estremamente negativo nell'insegnamento.

La storia del CAL, in effetti, non è incoraggiante. Nel 1980, ad esempio (quando nelle scuole britanniche esistevano appena 100 microcomputer), il mercato del software di-

dattico era pressoché inesistente. Gli insegnanti che avevano accesso ai computer erano costretti a redigere i propri programmi, in BASIC, senza magari avere la sufficiente esperienza di programmazione. Nella maggior parte dei casi, dunque, il software risultante era, a dir poco, rudimentale e molto elementare. Esistevano programmi che, pur essendo il frutto di ore e ore di programmazione da parte di un insegnante, si limitavano a visualizzare un «bravo» o un «ripeti» a seconda della risposta a una delle quattro operazioni aritmetiche. Tuttavia, anche programmi così banali ebbero inizialmente un discreto successo, poiché stimolavano l'interesse degli studenti più dei metodi tradizionali.

La ragione di un tale successo, ovviamente, non va ricercata nei programmi stessi, ma nel nuovo e affascinante mezzo adoperato: il computer. Ciò di cui molti non si resero conto era l'enorme costo occorrente per realizzare un sistema che, tutto sommato, equivaleva alla normale carta e penna.

In certi casi, per restare al passo coi tempi, alcuni insegnanti hanno accettato di usare software CAL di scarsissima qualità, solo perché era l'unico reperibile. Il risultato di tutto ciò è stato un conseguente "rigetto" di tutto ciò che "sapesse" di CAL.

Nel saggio *Young Learners and the Microcomputer* (Studenti giovani e microcomputer), David Chandler afferma, paradossalmente, che il microcomputer è un ottimo strumento per far compiere all'istruzione passi da gigante... per ritornare ai metodi del XIX secolo!

Sfortunatamente, la produzione commerciale di software CAL non può che avvalorare questa opinione negativa, rafforzando discutibili abitudini mentali e incoraggiando la mancanza di fantasia, poiché le risposte sono limitate a laconici "sì" o "no". Benché lo scopo originario del CAL fosse quello di insegnare a ciascun bambino in base al suo ritmo di apprendimento, spesso ci si è accorti che il ritmo era in realtà quello del computer.

Nonostante tutto, le sorti del CAL sembrano risollevarsi. Infatti, l'enorme diffusione di home computer (anche se adoperati soprattutto per videogiochi) tra giovani di età compresa tra gli 11 e i 18 anni ha creato un ampio "sottobosco" di hardware, utilizzabile anche per scopi più istruttivi.

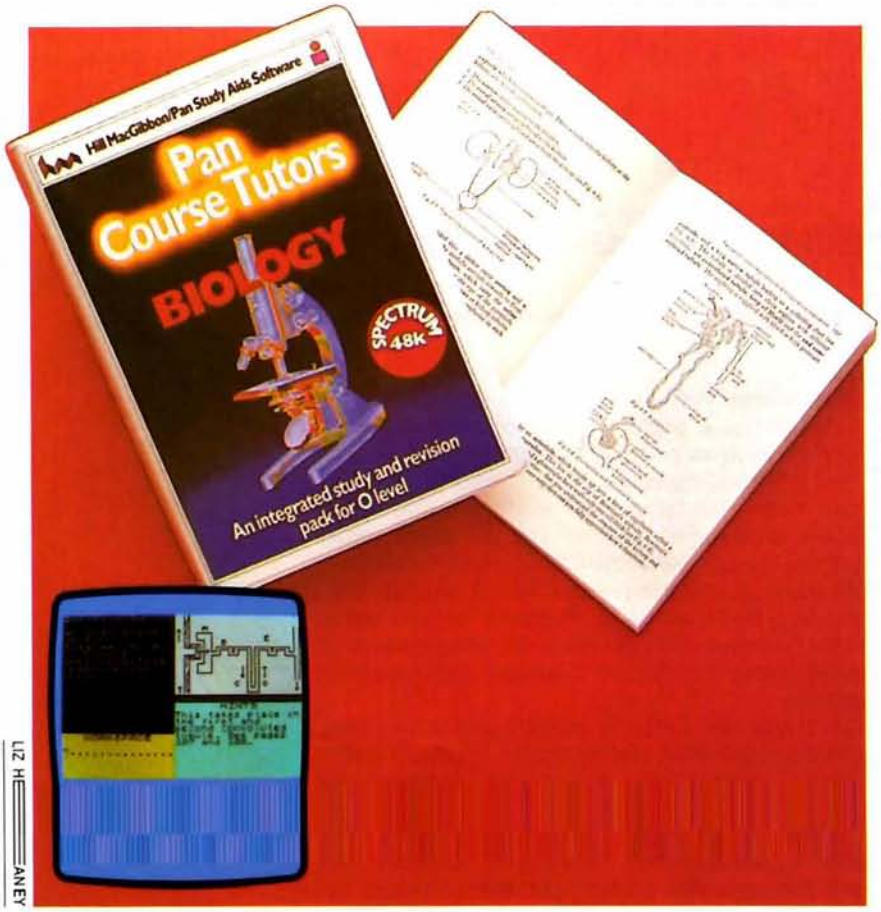
Creatasi questa premessa, la quantità del software CAL è andata notevolmente accrescendosi e anche la qualità è migliorata nel tentativo di offrire strumenti adatti allo studio "a casa".

In Gran Bretagna, paese all'avanguardia in questo settore, esistono grosse compagnie editoriali che hanno investito ingenti somme per la produzione di software didattico di supporto ai normali programmi scolastici o alla preparazione di esami.

Una di queste ditte è la Hill MacGibbon che, al pari delle concorrenti, ha individuato due aree principali d'intervento. La prima riguarda la produzione di package specifici destinati alla preparazione di esami, su particolari materie, e rivolti a singoli individui. La seconda area riguarda invece l'insegnamento in senso più lato, in cui il computer può servire per accostarsi ad eventi simulati, altrimenti non accessibili a causa dell'alto costo o della pericolosità.

Un buon esempio dei prodotti della prima categoria è la serie di programmi *Pan Course Tutors*, pubblicato congiuntamente dalla Pan Books e dalla Hill MacGibbon. Questa serie comprende sei titoli (*Matematica, Biologia, Chimica, Economia, Francese e Fisica*), ciascuno accompagnato da un libro di testo. La presenza del libro evita che il programma venga appesantito da enormi quantità di testo, lasciando al software una funzione più concentrata sull'aspetto interattivo dell'apprendimento.

La combinazione testo/software, valida per il momento, sembra però destinata ad un successo minore in futuro, quando la maggior disponibilità di memoria degli apparecchi renderà superflua, almeno in parte, la consultazione della carta stampata. Infatti, la soluzione che verrà adottata sarà probabilmente basata sull'uso di "finestre" sullo schermo, nelle quali far apparire il testo contemporaneamente all'applicazione del software.



Idee skinneriane

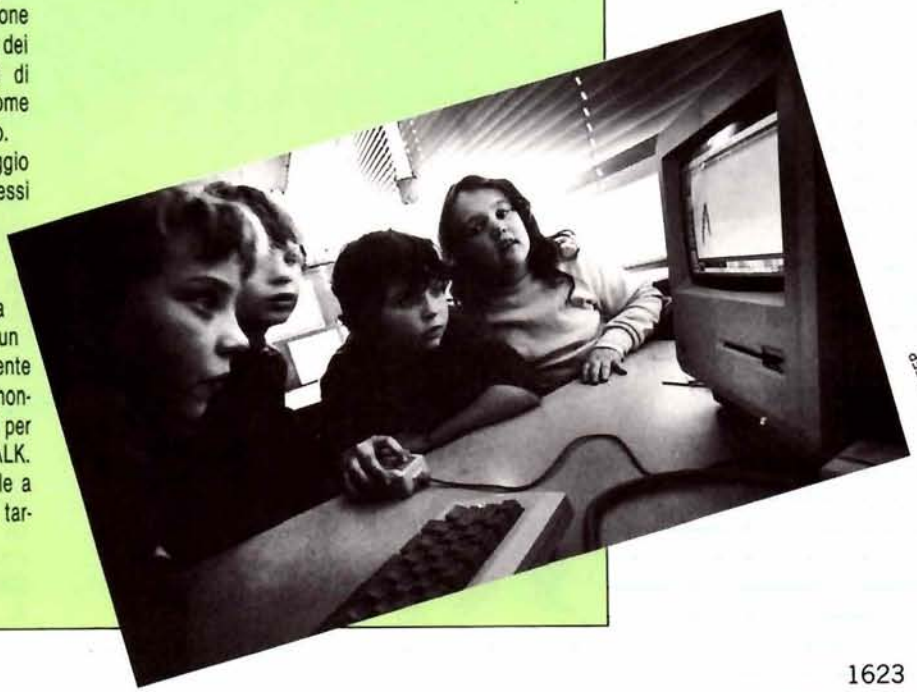
La serie "Pan Course Tutors", in cui il software è combinato con l'uso di libri di testo, è un tentativo di creare un nuovo rapporto tra materia studiata e studente. Tuttavia, anche se si possono considerare ottimi strumenti per il "ripasso", simili prodotti si richiamano ancora ai concetti primordiali del CAL e a quelli di B.F. Skinner

Nella serie dei *Pan Course Tutors*, i programmi rivolgono domande all'allievo e uno dei vantaggi, rispetto ad un normale libro, è che non si può andare a sbirciare la risposta qualche pagina più avanti. Se la risposta ad un quesito non è esatta, il programma talvolta fornisce qualche suggerimento (del tipo «Ti sei ricordato delle parentesi?»). Se anche la seconda risposta è sbagliata, viene data un'ulteriore traccia, in genere un riferimento ad un paragrafo del libro di testo, da consultare prima di proseguire. Ogni risposta, assieme al tempo impiegato a fornirla, viene registrata dal computer: ciò permette di formulare una valutazione completa di suggerimenti sulle parti della materia che conviene ripassare, prima di andare oltre. Un simile sistema consente veramente di studiare seguendo il proprio ritmo, alternando l'uso del computer a quello dei libri di testo, senza particolari pressioni sull'allievo.
Inoltre, il CAL può essere adoperato per insegnare ai più giovani ad usare il computer come strumento. Un programma che illustri gradualmente allo studente le varie funzioni offerte da un word processor, ad esempio, può avere un alto valore educativo: è certamente più facile seguire quanto avviene su di uno schermo, dopo aver premuto un tasto, che non cercare spiegazioni "statiche" in un manuale d'uso. Sul Macintosh della Apple, ad esempio, viene adottato un metodo simile per fornire al principiante informazioni sul sistema in generale e sui vari programmi utilizzabili in particolare.
Come effetto secondario di questo rinnovato interesse per il CAL, si assiste al rifiorire di prodotti software di scarso valore, il cui unico scopo (talvolta non sufficientemente evidente!) è quello di arricchire gli autori. Nonostante le presentazioni altisonanti, in cui si declamano le incredibili doti didattiche di un package, spesso i programmi risultano essere nient'altro che giochetti puerili con ben poco di educativo.
Una volta affievolitesi le reazioni suscitate dall'impatto dell'uso dei microcomputer, specialmente nell'ambito scolastico, si potrà forse esaminare con occhio più critico l'effettiva utilità e i precisi limiti di questi apparecchi, quali ausilio nell'insegnamento, particolarmente quello a livello individuale.

LE PROSPETTIVE

L'attuale produzione di software CAL è ancora condizionata da criteri e metodi classici d'insegnamento: non viene stimolata abbastanza la fantasia o la creatività dell'operatore (ossia lo studente). Affinché il CAL possa divenire, un domani non troppo lontano, un efficace strumento sono necessari tre cambiamenti radicali.
Per prima cosa va assolutamente eliminato l'antiquato metodo delle prove pratiche di apprendimento, così come è concepito e applicato dalla maggior parte dei programmi CAL ora esistenti. Poi sono necessari computer con una maggiore capacità di memoria (gli attuali microcomputer non sono in grado di gestire programmi interattivi di vaste dimensioni). L'ultimo cambiamento, forse quello più importante, è affidare lo sviluppo dei programmi a persone davvero competenti: fino ad ora, infatti, la maggior parte dei programmi per CAL sono stati realizzati da esperti di programmazione, che non sono sempre al corrente di come effettivamente si svolga il processo dell'apprendimento.
Uno dei primi tentativi in questo senso è stato il linguaggio PILOT, che facilitava la redazione, da parte degli stessi insegnanti, di programmi interattivi destinati agli allievi; ricerche in questo settore vengono tuttora condotte nel centro di ricerca della Xerox, a Palo Alto in California (USA), dove sono nati i concetti di base per il sistema operativo del Macintosh della Apple. Qui è stato creato un "Mondo Sperimentale", basato su software ampiamente visuale. In un simile ambiente software, anche i non-programmatori riescono a produrre programmi didattici per computer, usando uno speciale linguaggio, lo SMALLTALK. Questo "Mondo Sperimentale", regolato in modo simile a quanto avviene con il LOGO e la sua grafica con le tartarughe, richiede oggi una vasta quantità di memoria, ma rappresenta una tendenza che verrà presto seguita nell'ambito del CAL.
Le grandi potenzialità di sistemi basati su microcomputer e su apparecchi a videodischi, una volta messi a punto i metodi più adatti per facilitare l'interazione tra l'insegnante, lo studente e le macchine, potranno essere a livello mondiale creando, al tempo stesso, enormi banche dati accessibili a qualsiasi istituto o scuola. Il CAL, dunque, sembra avere un futuro molto più roseo di quanto si potersse prevedere dai suoi inizi costellati da incertezze, da tentativi falliti e da una generale diffidenza

Orizzonti nuovi
La diffusione di sistemi operativi corredati da un'efficace grafica (come quello creato per il Macintosh della Apple e il GEM della Digital Research) consente l'uso del computer anche a persone ignare del funzionamento dei suoi processi interni. Questo atteggiamento amichevole del software di base favorisce lo sviluppo di programmi CAL molto più sofisticati di quelli realizzati in passato


TONY SLEEP

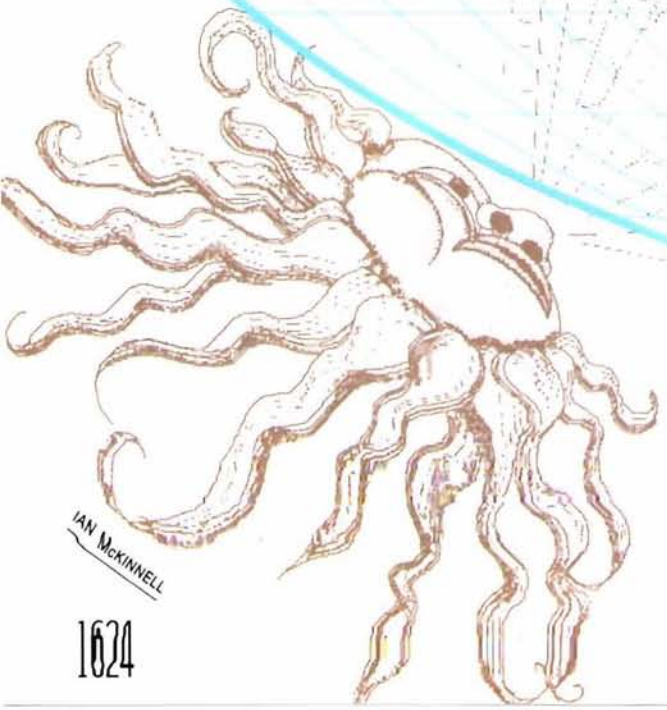
Uomo in mare!

Durante il viaggio accadono eventi imprevisti: ecco una prima routine che li genera a caso

Per simulare le difficoltà e gli incidenti quotidiani che capitavano durante i viaggi per mare del XVI secolo, occorre introdurre nel programma qualche elemento di tipo casuale. Alcuni eventi possono anche essere positivi (un buon vento di poppa), ma altri possono arrivare perfino a compromettere il buon esito della spedizione.
È previsto un totale di 16 eventi diversi, ma soltanto due di questi possono accadere in una sola settimana. Qui di seguito viene descritta la routine che seleziona gli eventi casuali e viene presentato il codice per 5 di essi. Prima di andare oltre, si rimuova (se inserita) la sezione tra le linee 601 e 604. Il modulo qui presentato è composto da più parti: una routine di inizializzazione, due linee da inserire nel corpo principale del programma e una subroutine per la selezione casuale degli eventi.







È importante che un particolare incidente capiti una sola volta durante il viaggio, pertanto a tal scopo viene dimensionata una matrice, RR(), alla linea 42. Questa contiene 16 elementi, che corrispondono ai 16 eventi imprevisti. Inizialmente ogni elemento è posto a 0 ma, quando si verifica un evento, l'elemento corrispondente è posto a 1, cosicché il programma è in grado di sapere se il particolare incidente si è già verificato o no. Oltre alla matrice, serve anche un contatore di eventi, per evitare che il programma vada inutilmente a ricercarne uno nuovo, quando tutti e sedici sono già esauriti. A questa funzione è preposta la variabile RM (si noti che il suo valore andrà in seguito variato, quando si aggiungeranno ulteriori eventi).
La linea 860, nel ciclo principale, contiene una chiamata alla routine generatrice di eventi casuali, situata alla linea 5500. La collocazione di questa chiamata segue la visualizzazione del Diario del Capitano e la stima della durata del viaggio.
La subroutine alla linea 5500 seleziona quale evento accade in una particolare settimana, ma esegue una semplice Return, qualora gli eventi siano esauriti. Questo fatto è controllato confrontando, alla linea 5502, il numero di eventi accaduti, in RC, col numero di eventi previsti dal programma, in RM.

La routine degli eventi

La linea 5510 genera un numero casuale compreso tra 1 e il massimo numero di eventi previsti (RM). La linea 5520 controlla la matrice RR(), per vedere se il corrispondente evento si è già verificato. Se trova un valore 1 la routine torna al programma chiamante senza ulteriori azioni. In caso contrario, la linea 5523 pone il valore dell'elemento a 1, per evitare che l'evento venga nuovamente riselezionato. La stessa linea provvede ad incrementare di uno il numero di eventi accaduti, in RC.
Alla linea 5525 una ON X GOTO provvede a diramare l'esecuzione in base all'evento scelto. Le strutture ON... GOTO e ON... GOSUB servono per evitare una lunga serie di IF... THEN, quando la scelta del percorso è multipla. Il comando è seguito da una serie di numeri di linea, a cui salta l'esecuzione, in base al valore della variabile X. Se X=1 si passa al primo numero di linea, se X=2 al secondo e così via. Si noti che lo Spectrum non prevede le strutture ON... GOTO e ON... GOSUB (consultare il riquadro "Basic & Basic").

A ciascuno dei cinque eventi possibili (per il momento), corrisponde la visualizzazione di una frase. Il primo evento, una persona fuori bordo, è descritto nella linea 5540. I numeri di linea che seguono, nell'elenco della ON... GOTO, rimandano tutti alla linea 5570, dove inizia la subroutine per la perdita di viveri e scorte, che corrisponde agli eventi 2, 3, 4 e 5.
Se il numero casuale è 1, dopo il messaggio che comunica la perdita di un uomo, la linea 5554 visualizza il nuovo totale dell'equipaggio, sottraendo 1 a CN. Tuttavia, la variabile CN non viene qui modificata: ciò verrà fatto in seguito, al termine del rapporto settimanale, quando viene verificato se il rendimento dei vari membri è pari a -999.
Ovviamente, va anche scelto quale membro dell'equipaggio far cadere in mare: di ciò si occupa la linea 5558, che scorre la matrice dell'equipaggio, alla ricerca del primo elemento vivente (diverso da 0 o da -999). La linea 5560 getta l'uomo a mare, assegnando il valore -999 al suo rendimento. A questo punto la variabile T è posta a 16, per uscire dal ciclo.
Gli altri eventi trattati in questa sezione, da 2 a 5 nella matrice RR( ), riguardano la perdita di uno tra i quattro tipi di scorte. Per vivacizzare il gioco, la quantità di scorte perse in mare varia di volta in volta. La quantità di frutta, verdura, carne e acqua è rappresentata, rispettivamente, da PA(1), PA(2), PA(3) e PA(4).
Alla linea 5572 viene sottratto 1 da X (il numero dell'evento), in quanto l'evento 1 è trattato a parte. Il numero casuale risulta adesso compreso tra 1 e 4, e viene fatto corrispondere a uno dei tipi di scorte alimentari. La linea 5574 controlla che una particolare scorta non sia esaurita, prima di comunicare al giocatore il nefasto incidente.
L'ammontare della perdita può essere di un terzo, di un quarto o di un quinto della scorta esistente. La frazione viene scelta a caso dalla linea 5592, dividendo la quantità di scorta rimasta, PA( ), per tre, quattro o cinque e sottraendo il risultato dalla quantità della scorta rimanente con l'equazione alla linea 5594.
Dopo aver aggiunto anche questo modulo si può provare il gioco: il viaggio risulterà certamente più difficile. Nella prossima parte vengono aggiunti, per fortuna, eventi positivi.

Basic & Basic

Spectrum:

Inserire le seguenti modifiche:

```
5504 CLS
5525 IF X=1 THEN GOTO 5540
5526 IF X>1 AND X<6 THEN GOTO 5570
5535 LET I$ = INKEY$: IF I$ ="" THEN GOTO 5535
```

Micro BBC:

Apportare le seguenti modifiche:

```
5504 CLS
5535 I$ = GET$
```

MODULO SEI: EVENTI CASUALI

Inizializza le variabili per gli eventi

```
42 DIM RR(16)
43 REM INDICATORE DI EVENTO CASUALE
   GENERATO O NO
44 RC = 0
45 REM CONTATORE DI EVENTI CASUALI
46 RM = 5
47 REM NUMERO DI EVENTI CASUALI CONSENTITI
```

Aggiunta al programma principale

```
860 GOSUB 5500
861 REM CHIAMATA PER GENERARE EVENTI CASUALI
```

Subroutine per gli eventi casuali

```
5500 REM GENERATORE DI EVENTI CASUALI
5502 IF RC = RM THEN RETURN
5503 REM ESCE SE TUTTI GLI EVENTI SONO GIA'
     ACCADUTI
5504 PRINT CHR$(147)
5505 GOSUB 9200
5510 X = INT(RND(1)*RM) + 1
5515 REM GENERA UN NUMERO CASUALE TRA 1 E RM
5520 IF RR(X) = 1 THEN RETURN
5522 REM ESCE SE EVENTO GIA' USATO DAL
     PROGRAMMA
5523 RR(X) = 1:RC = RC + 1
5524 REM MODIFICA L'INDICATORE DI EVENTO E
     AGGIORNA IL CONTATORE DI EVENTI
5525 ON X GOTO 5540, 5570, 5570, 5570, 5570
5528 REM SALTO ALL'APPROPRIATA SEZIONE DI
     PROGRAMMA
5530 PRINT:S$ = K$:GOSUB 9100
5535 GET I$:IF I$ ="" THEN 5535
5539 RETURN:REM TORNA ALLA ROUTINE PRINCIPALE
5540 REM EVENTO 1 - UOMO A MARE!
5542 PRINT
5543 S$="DURANTE LA SETTIMANA*":GOSUB 9100
5545 PRINT:GOSUB 9200
5546 S$=" 1 PERSONA E' FINITA IN MARE*":
     GOSUB 9100
5548 S$=" DURANTE UNA BURRASCA*":GOSUB 9100
5550 PRINT:GOSUB 9200
5552 S$="L'EQUIPAGGIO E' ORA RIDOTTO A*":
     GOSUB 9100
5554 PRINT CN-1;"MEMBRI"
5558 FOR T = 1 TO 16
5559 REM CERCA MEMBRO DA ELIMINARE
5560 IF TS(T,2) = 0 OR TS(T,2) = -999
     THEN 5566
5562 TS(T,2)=-999:REM MEMBRO MORTO
5564 T = 16
5566 NEXT
5568 GOTO 5530
5570 REM EVENTI 2-5 - PERDITA DI SCORTE
5572 X=X-1:REM ORA X PUNTA A SCORTA (1-4)
5574 IF PA(X) = 0 OR PA(X) = -999 THEN 5530
5576 REM NESSUNA AZIONE SE SCORTA
     GIA' ESAURITA
5578 PRINT
5580 S$="DURANTE LA SETTIMANA*":GOSUB 9100
5582 PRINT:GOSUB 9200
5584 PRINT "UNA PARTE DI ";P$(X)
5586 S$="E' FINITA IN MARE*":GOSUB 9100
5588 PRINT:GOSUB 9200
5590 S$="NE RESTA PER CIRCA*":GOSUB 9100
5592 PA(X) = PA(X) - INT(PA(X)/(INT(RND(1)*3)+3))
5593 REM RIDUCE SCORTA DI 1/2 1/3 O 1/4
5594 PRINT INT(PA(X)/(CN*PN(X)));
5595 PRINT "SETTIMANE"
5599 GOTO 5530
```

Le matrici in PASCAL

Un'alternativa alle strutture dati molto più complesse del PASCAL

In molti linguaggi di programmazione l'impiego di matrici, per conservare dati sotto forma tabulare, diventa un fatto naturale e scontato. Tuttavia la principale ragione per l'uso quasi universale delle matrici deriva, molto spesso, da una pura e semplice abitudine o, in certi casi, dalla mancanza di altri tipi di strutture per i dati, caratteristica tipica dei linguaggi meno recenti: nel FORTRAN, progenitore di molti altri linguaggi di alto livello, esistono, ad esempio, numeri complessi e matrici. Nella formulazione del BASIC sono stati tralasciati i numeri complessi, mantenendo però le matrici.
Ciò poteva andar bene quando il FORTRAN veniva impiegato per calcolare individualmente ogni elemento di una matrice e quando il BASIC serviva da linguaggio propedeutico all'apprendimento del FORTRAN, ma oggi le cose sono ben diverse.
Si è già vista la versatilità delle strutture condizionali WHILE e REPEAT offerte dal PASCAL e si è anche sottolineata l'enorme potenzialità derivante dalle numerose strutture di dati presenti in questo interessantissimo linguaggio. Non c'è da stupirsi, dunque, se l'impiego delle matrici, benché previsto dal PASCAL, passi in secondo piano.

Versatilità delle matrici in PASCAL

Prima di esaminare l'uso delle matrici è bene insistere sul fatto che molti problemi in PASCAL possono essere risolti, quasi sempre, in modo più brillante ricorrendo ad altre strutture di dati, anziché fossilizzarsi, come fanno alcuni programmatori, sul terreno familiare delle matrici. Dopo questa premessa, non si deve però credere che il PASCAL ostacoli l'uso delle matrici: al contrario, sfruttando la maggior flessibilità delle strutture dei dati, ne amplia l'applicazione.
La definizione di una matrice, nell'apposita sezione TYPE, serve per tre distinti motivi: accantona sufficiente memoria per contenere gli elementi di lunghezza fissa che la compongono e definisce sia il "tipo" del suo indice che quello di ogni suo componente. Così, per esempio:

```
TYPE
    ContoCar = ARRAY ['A'..'Z'] OF integer;
VAR
    lista : ContoCar;
```

riserva spazio sufficiente per 26 interi, ai quali si può far riferimento usando l'identificatore della matrice, seguito da un'espressione, racchiusa tra parentesi quadre, che stabilisce un particolare indice. Detto per inciso, anche le prime versioni di BASIC adoperavano le parentesi quadre per le matrici, ma in seguito vennero sostituite con le tonde. Per accedere a un particolare elemento della matrice definita, dunque, si può scrivere:

```
lista ['M'] oppure lista [pred (simbolo) ]
```

Nel secondo caso il valore dell'espressione pred(simbolo) deve corrispondere, ovviamente, a un valore di tipo char, compreso tra 'A' e 'Z'. Come indice in una matrice può essere usato qualsiasi scalare, ma non tipi appartenenti ai numeri reali o a strutture complesse. Ciò impedisce di usare espressioni prive di senso del genere lista ['secondo'] o flag [3.45], per esempio. Adoperando la sezione TYPE già vista, si può inizializzare (a 0 in questo caso) il valore di ciascun elemento della matrice, con:

```
VAR
    lettera : char;
    contatore : ContoCar;
BEGIN
    FOR lettera := 'A' TO 'Z' DO
        contatore[lettera] := 0
```

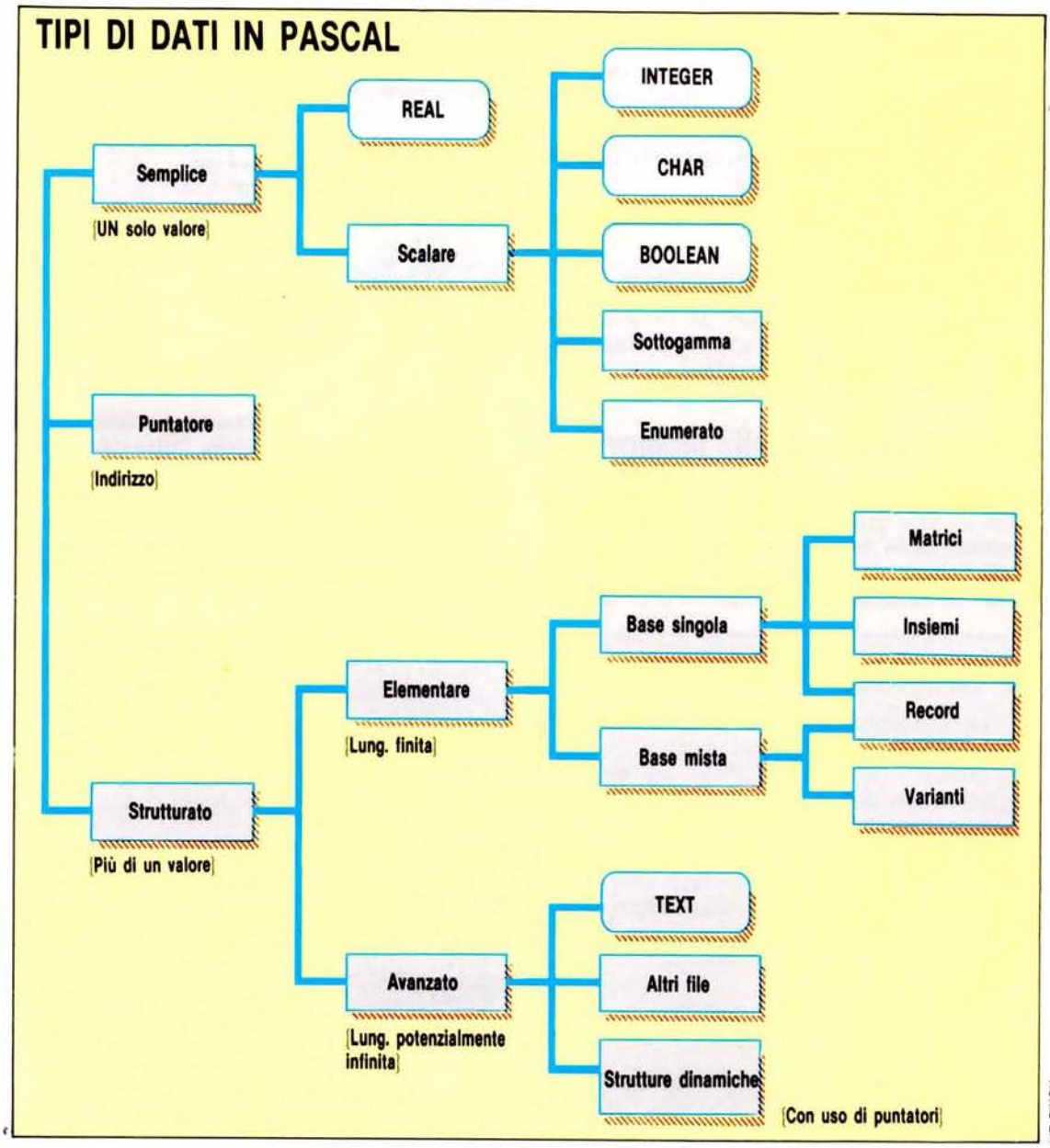
Ovviamente, considerata la definizione della matrice, contatore [1] non è un'espressione valida (l'indice appartiene al tipo sbagliato) e contatore ['a'] non esiste, poiché la gamma degli indici prevede soltanto le lettere maiuscole. Pertanto conviene sempre definire adeguatamente il tipo degli identificatori usati come indici.
Per rendere una sezione di programma maggiormente adattabile, si può riscrivere la definizione, in modo da specificare la sottogamma di valori nella sezione CONST: a questo punto basta modificare una sola linea di programma per adattarlo a diverse circostanze d'uso:

```
CONST
    LungStringa = 25;
TYPE
    DimStringa = 1..LungStringa;
    stringa = PACKED ARRAY [DimStringa] OF char;
```

Si noti che la parola riservata PACKED può precedere qualsiasi parola identificativa di tipi strutturati (SET, ARRAY, RECORD o FILE). Usando il formato PACKED ("impaccato") si ha la possibilità di risparmiare notevoli quantità di memoria, particolarmente utile su apparecchi a 16 o 32 bit.

Il "tipo" stringa

L'unico caso in cui occorre prestare veramente attenzione, con il formato PACKED, è quando



Tipologia... tipica!

La strutturazione molto disciplinata dei dati, in PASCAL, costringe il programmatore ad una oculata e metodica analisi durante la realizzazione di un programma. Solo due dei tre tipi basilari di dati, qui illustrati, possono essere suddivisi in ulteriori categorie. Ma la gamma di possibili dati semplici e strutturati implica una ben precisa definizione di tutti i dati adoperati in un programma. Questo notevole sforzo, da parte del programmatore, viene ricompensato da una maggiore eleganza nella soluzione di problemi di programmazione rispetto ad altri linguaggi

si usano costanti di stringhe letterali. Una stringa di caratteri racchiusa tra apici è considerata come una matrice il cui indice varia da 1 (attenzione: non da 0) alla lunghezza massima. Ciò significa che, implicitamente, essa è dichiarata come:

```
PACKED ARRAY [1..N] OF char
```

in cui N è il numero di caratteri nella stringa. Si consideri il programma:

```
PROGRAM ImpaccaStringa;
CONST
    Pascal = 'Pascal';
TYPE
    stringa = PACKED ARRAY [1..10] OF char;
VAR
    S : stringa;
BEGIN
    S := Pascal;
    S := 'Troppi caratteri';
    S := 'Pascal '
END.
```

Nel corpo del programma, le prime due linee non sono valide, in quanto incompatibili in lunghezza, mentre la terza è accettabile, poiché vengono usati quattro spazi per riempire le posizioni rimanenti nella stringa. Se la definizione del tipo "stringa" non fosse PACKED, qualsiasi assegnazione a elementi letterali sarebbe incompatibile. In pratica queste restrizioni non implicano grossi problemi. Benché, ufficialmente, il PASCAL preveda soltanto l'uso delle procedure read e write per l'ingresso/uscita di intere strutture da e per i file, è consentita l'emissione di letterali o variabili stringa di questo tipo anche direttamente sullo schermo. Si scriva un programma con la definizione di stringa data nell'ultimo esempio, e si aggiungano le linee:

```
REPEAT
    write ('Stringa (Q per finire):');
    ReadLn (S);
    WriteLn ('La stringa era:"',s,'"')
UNTIL S[1] IN ['Q','q']
```

Reso il programma funzionante, lo si provi e si verifichino i seguenti punti:

- Gli spazi e le tabulazioni iniziali vengono ignorate?
- Che cosa accade se la linea immessa supera la lunghezza della stringa?
- Che cosa contiene s se viene immesso un solo carattere?
- E se viene premuto soltanto il Return?
- Si è capaci di aggiungere al programma una parte che provveda a riempire con spazi gli elementi insignificanti?

Limitazioni dovute alla memoria

Per quanto riguarda il PASCAL non esistono limiti né alla grandezza né al numero di dimensioni delle matrici. Quindi, se a qualcuno venisse in mente la balzana idea di andare a caccia di Klingons in uno spazio a 17 dimensioni, in PASCAL esiste la struttura di dati adatta per farlo! Tuttavia, il proprio computer va perdonato se non accetta una dichiarazione di questo genere:

```
TipoEnorme = ARRAY [integer] OF SET OF char;
    {a richiesta di almeno 64 Kbyte x 128!}
Enormissimo = ARRAY [1..1000] OF
    RECORD
        cognome,
        nome : string;
        indirizzo : ARRAY [1..5] OF string;
        presente : SET OF presenze;
        {ecc.}
    END; {enormissimo}
```

I limiti di memoria mettono presto fine a simili e ambiziosi voli di fantasia. Tuttavia, in successive parti di questa serie, viene spiegato come far sì che soltanto una minima parte degli elementi di una struttura possa essere caricata in memoria, lasciando il resto su dischetto: un'altra caratteristica impagabile del PASCAL!

IL SETACCIO DI ERATOSTENE

In questo noto programma da "banco di prova", per l'individuazione di numeri primi, la velocità d'esecuzione è ottimizzata ignorando i numeri pari e consultando una matrice di flag (8192 elementi per i numeri primi fino a 16384). Una simile matrice occupa almeno 8 Kbyte di memoria, se non addirittura oltre 32 Kbyte nel caso il compilatore non preveda variabili booleane a singolo byte e interi a quattro byte. Ovviamente, perché un simile programma possa girare su di un home computer, va adeguatamente modificato.
L'algoritmo adottato, comunque, è il seguente:

```
Mette tutti i numeri (1...max) in un setaccio
Estrae il numero uno (numero primo per definizione)
{lo visualizza se necessario}

RIPETI (REPEAT)
    Estrae il più piccolo numero nel setaccio
    {visualizza questo numero primo}
    rimuove tutti i suoi multipli dal setaccio
FINCHÈ (UNTIL) il setaccio è vuoto
```

Un insieme così ampio non rientra nella limitata memoria di un home computer, tuttavia si può applicare l'algoritmo riducendo le dimensioni della matrice degli insiemi (a 100 o a 1000 elementi, a seconda del compilatore usato). Ogni indice per la matrice di insiemi si ricava dalla divisione intera del numero e la sua appartenenza all'insieme è rappresentata da N modulo 100 (o 1000).
Tuttavia, questa versione ridotta è piuttosto lenta rispetto a quella che funziona con la matrice

```
PROGRAM         ListaSetaccio      (output);
{Trova i numeri primi applicando l'algoritmo
 di Eratostene}
CONST
   DimIns       =    100; {^implementazione}
   PredDimIns   =     99; {DimIns - 1}


   MaxPrimo     = 16383; {per MaxInt = 32767}
   ListMax      =   163; {MaxPrimo DIV DimIns}

TYPE
   GammaPrimi   = 1..MaxPrimo;
   dimensione   = 0..ListMax;
   GammaIns     = 0..PredDimIns;
   Siv          = SET OF GammaIns;
   Eratostene   = ARRAY [dimensione] OF Siv;
   cardinale    = 0..MaxInt;

VAR
   Setacci      : Eratostene;
   conto,
   multiplo     : cardinale;
   indice       : dimensione;
   N            : GammaPrimi;
   numero       : 0..PredDimIns;

BEGIN
   WriteLn;
   WriteLn ('Setaccio di Eratostene' : 50);
   WriteLn ('======================' : 50);
   WriteLn;
   WriteLn;

   FOR indice := 0 TO ListMax DO
     Setacci [indice] := [0..PredDimIns];
         {mette ogni numero nel setaccio}

   Setacci [0] := [2..PredDimIns];
{WriteLn (1);}{numero primo per definizione}
   conto := 1;

   FOR N := 2 TO MaxPrimo DO
     BEGIN
       indice := N DIV DimIns;
       numero := N MOD DimIns;

       IF numero IN Setacci [indice] THEN
         BEGIN
           conto := succ (conto);
         {WriteLn (N);}
           Setacci [indice] := Setacci [indice]
                               - [numero];
           multiplo := N+N;

           WHILE multiplo <= MaxPrimo DO
             BEGIN
               indice := multiplo DIV DimIns;
               Setacci [indice] := Setacci [indice]
                       - [multiplo MOD DimIns];
               multiplo := multiplo + N
             END

         END

     END;

   WriteLn;
   WriteLn (conto : 25, ' = numeri primi trovati.');

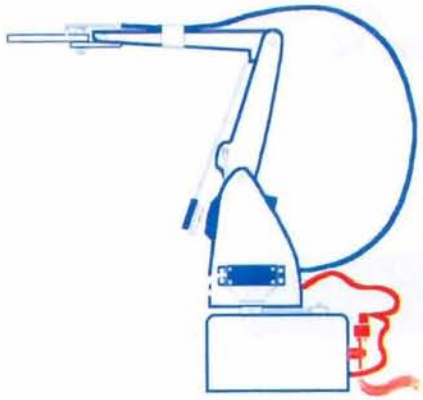
END.
```

DATABASE

La seconda tabella della mappa di memoria del Commodore 64. Per concessione della Commodore Business Machines

INDIRIZZO ESADECIMALE	ETICHETTA	LOCAZIONE DECIMALE	DESCRIZIONE
0016	TEMPPT	22	**Pointer: Temporary String Stack** Puntatore: Stack provvisorio per le stringhe
0017-0018	LASTPT	23-24	**Last Temp String Address** Ultimo indirizzo provvisorio di stringa
0019-0021	TEMPST	25-33	**Stack for Temporary Strings** Stack per stringhe provvisorie
0022-0025	INDEX	34-37	**Utility Pointer Area** Area per puntatori di sistema
0026-002A	RESHO	38-42	**Floating-Point Product of Multiply** Risultato di moltiplicazione in virgola mobile
002B-002C	TXTAB	43-44	**Pointer: Start of BASIC Text** Puntatore: Inizio del testo in BASIC
002D-002E	VARTAB	45-46	**Pointer: Start of BASIC variables** Puntatore: Inizio dell'area per le variabili BASIC
002F-0030	ARYTAB	47-48	**Pointer: Start of BASIC Array** Puntatore: Inizio dell'area per le matrici BASIC
0031-0032	STREND	49-50	**Pointer: End of BASIC Arrays (+1)** Puntatore: Termine area delle matrici BASIC (+1)
0033-0034	FRETOP	51-52	**Pointer: Bottom of String Storage** Puntatore: Fondo dell'area riservata alle stringhe
0035-0036	FRESPC	53-54	**Utility String Pointer** Puntatore a stringhe di utilità
0037-0038	MEMSIZ	55-56	**Pointer: Highest Address Used by BASIC** Puntatore: Indirizzo più alto usato dal BASIC
0039-003A	CURLIN	57-58	**Current BASIC Line Number** Numero di linea BASIC corrente
003B-003C	OLDLIN	59-60	**Previous BASIC Line Number** Numero di linea BASIC precedente
003D-003E	OLDTXT	61-62	**Pointer: BASIC Statement for CONT** Puntatore: per l'istruzione CONT del BASIC
003F-0040	DATLIN	63-64	**Current DATA Line Number** Numero di linea corrente per le DATA
0041-0042	DATPTR	65-66	**Pointer: Current DATA Item Address** Puntatore: Indirizzo del corrente elemento di DATA
0043-0044	INPPTR	67-68	**Vector: INPUT Routine** Vettore: Routine d'immissione (INPUT)
0045-0046	VARNAM	69-70	**Current BASIC Variable Name** Nome di variabile BASIC corrente

Fine di un braccio



Le connessioni elettriche

Fase 1: Il collegamento della scheda

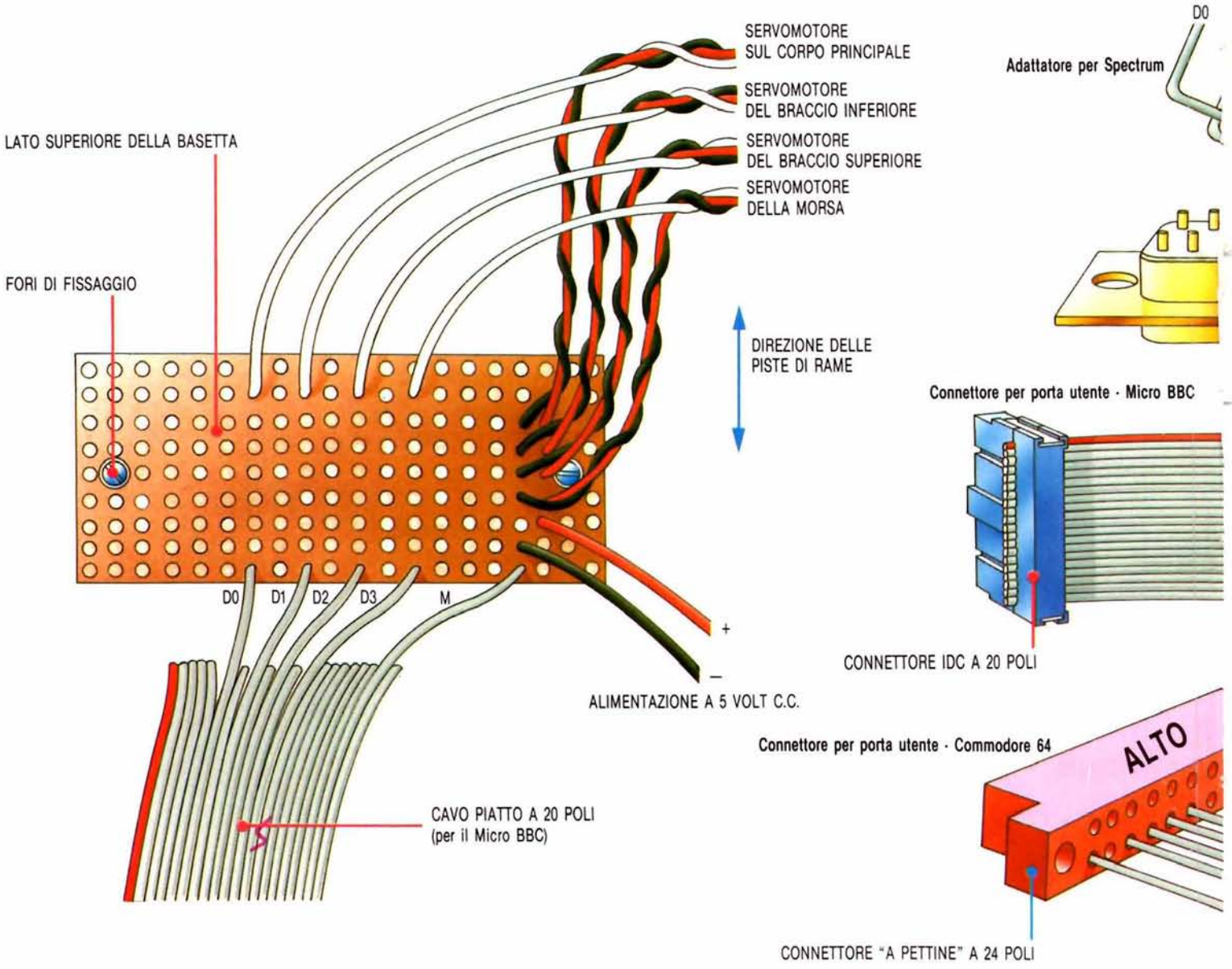
Ciascun servomotore possiede tre fili di collegamento. Sul modello Futaba, adottato nel prototipo, questi tre fili sono così colorati: bianco per la linea di controllo, rosso per il positivo dell'alimentazione e nero per il negativo e il ritorno di massa. Per collegare i cavetti dei servomotori al cavo piatto che porta al computer, conviene utilizzare una basetta traforata tipo veroboard, con 20 piste per 9 fori. Questa basetta viene in seguito fissata all'esterno della scatola principale. Prima di saldare i collegamenti, occorre praticare sulla scheda due fori per il fissaggio, come in figura, e forare (usando la basetta come maschera) anche la scatola di base del braccio. Un altro foro, sempre sulla scatola di base, serve per il passaggio dei cavetti relativi ai motori contenuti nella base.

Si riuniscano adesso i quattro gruppi di fili provenienti dai servomotori e si saldino nelle posizioni indicate. Le quattro linee dei dati e la massa, provenienti dal computer, vanno saldati sulla parte inferiore della basetta. La figura mostra il cavo piatto a 20 poli usato con il Micro BBC. Per Spectrum e Commodore 64 si possono usare semplicemente 5 cavetti separati oppure 5 poli di un cavo piatto. Si noti che ogni linea di controllo (cavetto bianco) va saldata in corrispondenza delle linee dei dati, mentre tutti i cavetti neri (la massa) vengono uniti assieme e saldati alla linea di massa proveniente dal computer.

L'ultima operazione di saldatura riguarda i cavetti dell'ali-

FASE 1: IL COLLEGAMENTO DELLA SCHEDA

FASE 2: CONNESSIONE ALLA PORTA



mentazione a 5 volt. Poiché la linea a 5 volt, per nessuno dei tre computer, fornisce una corrente sufficiente ad alimentare i servomotori del braccio, occorre un'alimentazione separata. L'ideale sarebbe poter disporre di un alimentatore da 5 volt a corrente continua, ma si può anche usare una pila da 4,5 volt (o tre elementi da 1,5 volt in serie). In tal caso conviene saldare ai cavetti di alimentazione appositi contenitori per pile, già completi di cavetti di raccordo colorati (quello rosso corrisponde sempre al positivo). Il polo negativo per l'alimentazione dei motori è comune alla linea di massa del computer, mentre il positivo va collegato a ciascuno dei quattro cavetti rossi dei servomotori, tramite piste ramate.

Fase 2: Connessione alla porta del computer

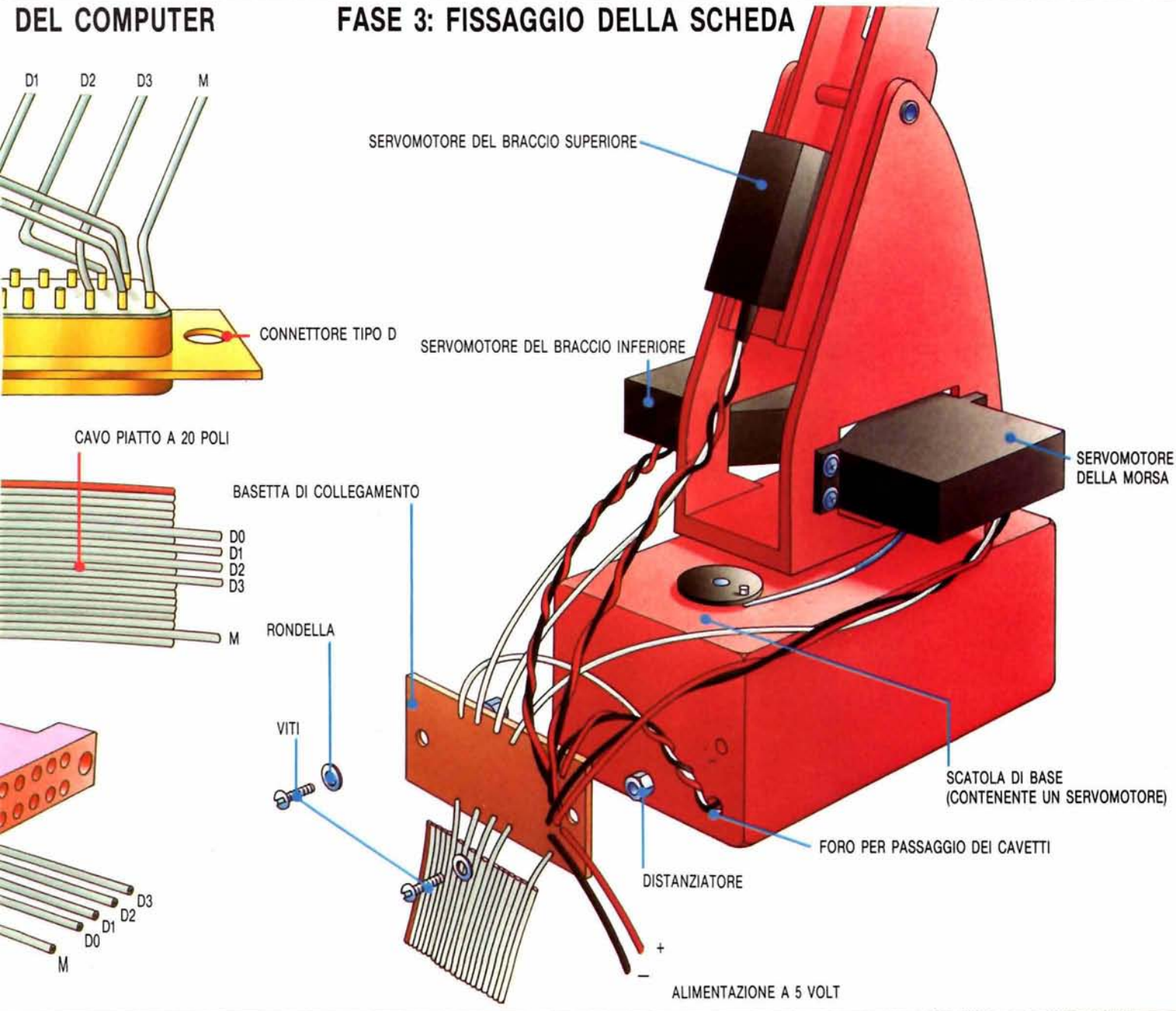
Completata la fase 1, va montato adesso il connettore appropriato all'estremità libera del cavo piatto, collegando le linee D0-D3 dei dati e quella di massa. Per il Micro BBC occorre un connettore IDC a 20 vie. Per il Commodore 64 si usa un connettore "a pettine" da 24 poli (siccome tale connettore può essere inserito nei due sensi, si marchi chiaramente qual'è la parte superiore, ad esempio con TOP, come in figura).
Sia il BBC che il Commodore 64 possiedono i circuiti occorrenti per il controllo di periferiche tramite la porta utente, a differenza dello Spectrum, per il quale è necessaria invece una speciale interfaccia (descritta alle pagg. 1472, 1492 e 1514). I collegamenti sul connettore per tale interfaccia sono riportati in figura.
Le linee per i dati e per l'alimentazione dell'interfaccia provengono da un connettore tipo D a 12 vie, da inserire direttamente sul robot realizzato in precedenza. Si può adattare tale cavo aggiungendovi un connettore tipo D a 15 vie, maschio e femmina, per usarlo sul braccio robot. Questo sistema permette di collegare il braccio robot alla porta d'espansione dello Spectrum, tramite la scheda d'interfaccia.

Fase 3: Fissaggio della scheda

Dopo aver eseguito e controllato attentamente tutti i collegamenti, la basetta può essere fissata lateralmente alla scatola di base del braccio robot, come indicato in figura, tramite due bulloncini e relativi dadi. Assicurarsi che i collegamenti dei servomotori siano nel corretto ordine. La linea dati D0 (quella più a sinistra) deve collegarsi con il cavetto bianco del motore sul corpo principale del braccio robot, montato nella base. D1 deve essere collegato al servomotore situato sulla parte inferiore del braccio. D2 controlla il servomotore posto sulla parte superiore del braccio (vicino alla "spalla") e D3 controlla il servomotore per la morsa (montato sulla destra del corpo principale del braccio).

DEL COMPUTER

FASE 3: FISSAGGIO DELLA SCHEDA



KEVIN JONES

Massima immissione

Versatilità di un DBM dalle grandi qualità: il "dBase II"

Il lavoro di un "DataBase Manager", o più brevemente DBM, va ben al di là della semplice catalogazione e archiviazione di dati. In genere l'accesso ai record avviene tramite semplici comandi, ma i DBM più sofisticati possiedono un vero e proprio linguaggio interno che consente la creazione di complesse procedure.
Ciò moltiplica le possibilità operative ed accresce in modo significativo l'utilità di un database.
Per comprendere la funzione delle procedure scritte dall'utente, occorre prima esaminare quali siano i comandi semplici normalmente messi a disposizione dalla maggior parte dei DBM. Comandi quali SEARCH, LOCATE, DISPLAY, NEXT, LAST ecc. (corrispondenti a CERCA, LOCALIZZA, VISUALIZZA, PROSSIMO, ULTIMO, ecc.) vengono interpretati dal DBM aprendo un particolare file di dati ed esaminando i record sequenzialmente, alla ricerca di quelli che corrispondono alle specifiche indicate dall'utente. Questi comandi vengono solitamente raggruppati sotto la dicitura "linguaggio d'interrogazione".
Tuttavia esiste una intera serie di potenti DBM in cui è consentito creare procedure di accesso molto più sofisticate e complesse. *L'Archive* della Psion (vedere a pag. 1604) e il *dBase II* della Ashton Tate ne sono due ottimi esempi. Dette procedure non sono sequenze di comandi semplici, bensì espressioni redatte in un linguaggio interno, concepito a questo scopo. Alcuni di questi linguaggi assomigliano ai normali linguaggi di programmazione e il loro uso è facile da imparare (quello di *Archive* è molto simile al *SuperBASIC* della Sinclair).
Il linguaggio del *dBase II* si discosta invece dalla norma, anche se può essere assimilato a un BASIC strutturato. Ci sono alcune parole riservate, facili da imparare, e alcune poderose strutture: DO WHILE...ENDDO, DO CASE...ENDCASE e IF...ENDIF. Inoltre, sono presenti più di cento comandi, alcuni dei quali familiari ai programmatori, altri specificamente concepiti per l'uso di un database. Ecco alcuni esempi: CALL (per chiamare una routine in codice macchina), NOTE (l'equivalente della REM in BASIC) e STORE <espressione> to <variabile> (equivalente del LET <variabile> = <espressione> in BASIC). Comandi più insoliti sono: SKIP (per spostarsi avanti o indietro tra i record di un file), PACK (per rimuovere da un file i record divenuti inutili) e FIND <stringa di caratteri> per trovare una particolare sequenza alfanumerica.
Sono inoltre incorporate alcune funzioni: anche qui alcune sono familiari a chi usa il BASIC, altre un po' meno. CHR <espressione> equivale a CHR$(x) del BASIC, LEN <stringa di caratteri> equivale alla LEN del BASIC, TYPE <espressione> fornisce indicazioni sul "tipo" cui appartiene l'espressione e DATE( ) contiene la data di elaborazione.
Per illustrare la maggiore utilità delle procedure di accesso complesse, rispetto ai comandi semplici, viene qui proposta la realizzazione di un archivio di magazzino, usando il linguaggio del *dBase II*. I record del file, in questo esempio, sono volutamente mantenuti ad un livello semplice e sono composti da quattro campi numerici e uno letterale. Un tipico record (usato anche in precedenza) è:

MATRICOLA DI FABBRICA	06116
CODICE DI LISTINO	3995
GIACENZA IN MAGAZZINO	85
PREZZO	12.25
DESCRIZIONE	Vite a testa piatta

Per prima cosa occorre stabilire un nome (composto da una sola parola) per ciascun campo, poi fissare la lunghezza di ogni campo e quindi indicare se si tratta di un campo numerico, alfabetico o "logico".
Ad esempio:

MATRICOLA	: 5 caratteri; numerico
CODICE	: 5 caratteri; numerico
GIACENZA	: 3 caratteri; numerico
PREZZO	: 6 caratteri; numerico
DESCRIZIONE	: 40 caratteri; alfabetico

Una volta lanciato il *dBase II*, sullo schermo compare un punto, a indicare che il programma attende un comando. Quello che avvia la creazione di un nuovo file è CREATE, al quale occorre abbinare un nome identificativo dell'archivio (nel nostro caso è stato scelto MAGAZZ). Ecco un esempio:

```
.create magazz <CR>
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD NAME, TYPE, WIDTH, DECIMAL PLACES

001     matricola,n,5,0
002     codice,n,5,0
003     giacenza,n,3,0
004     prezzo,n,6,2
```

```
005     descrizione,c,40
006     <CR>
```

I dati della descrizione, immessi in lettere minuscole, non possono superare i 40 caratteri. Si noti che per i campi numerici viene richiesto anche il numero di posti decimali dopo la virgola, oltre alla lunghezza del campo.
La pressione del tasto Return senza immettere informazioni (come alla linea 006) termina questa fase preliminare.
Adesso occorre inserire qualche dato e ciò avviene usando il comando APPEND, che provoca una pulizia dello schermo e la comparsa di un record "intestato", ma vuoto:

```
MATRICOLA     :          :
CODICE        :          :
GIACENZA      :    :
PREZZO        :          :
DESCRIZIONE   :
```

I dati si possono ora digitare opportunamente nei vari campi. Ciascuna immissione termina con la pressione di Return e, completata anche quella dell'ultimo campo, il processo riparte da capo per procedere all'immissione del record successivo. La fase di immissione viene conclusa premendo Return (senza altre immissioni) come risposta al primo campo di un record vuoto.
Introdotti i dati, si vorrà ora accedere ad essi in base a più criteri. Supponiamo che serva sapere il prezzo di vendita globale di tutta la giacenza di magazzino. Un sistema potrebbe essere quello di richiamare ad uno ad uno i record e annotarsi sia il prezzo sia la giacenza, moltiplicare i due valori e poi sommare tutti i risultati. Questo è l'unico metodo possibile con uno schedario manuale, ma non è il più adatto per un database. Ecco come ottenere lo stesso risultato con *dBase II*:

```
.use magazz
.sum giacenza * prezzo
37870.58
```

La prima linea comunica al *dBase II* che intendiamo usare l'archivio MAGAZZ, la seconda indica al programma che desideriamo sommare il prodotto di prezzo x giacenza di tutti i record esistenti. La terza linea è il risultato di tale operazione, nel nostro caso espresso in Lire. Come si vede, l'operazione è abbastanza elementare.
Questo esempio dimostra la potenza di un DBM, anche usando qualche semplice comando. Il meglio viene quando si passa a memorizzare in un file una sequenza di comandi, ossia un vero e proprio programma.

Un esempio di dialogo, necessario per conservare la sequenza appena vista in un programma chiamato VALORE, è il seguente:

```
.modify command
ENTER FILE NAME: valore
set talk off
use parts
sum giacenza * prezzo
set talk on
cancel
```

Questo breve programma viene automaticamente memorizzato su disco e può d'ora in poi essere richiamato lavorando in *dBase II* semplicemente con:

```
.do valore
```

che provoca il caricamento in memoria della definizione e la sua immediata esecuzione, al termine della quale il controllo ritorna al *dBase II*.

UNA PROCEDURA... CONTABILE!

```
* Elenco dei ragionieri
SET TALK OFF
USE SOCI
DO WHILE.NOT.EOF
    IF PROFESSIONE = "RAGIONIERE"
        DISPLAY NOME
    ENDIF
    SKIP
ENDDO
```

Il programma qui riportato, che estrae dall'archivio di un'associazione tutti i membri di professione "ragioniere" è scritto nel linguaggio interno del "dBase II" della Ashton Tate. La prima linea, che inizia con un asterisco, è di puro commento. Poiché non è richiesta la visualizzazione per intero di ogni record individuato, la seconda linea contiene la sequenza SET TALK OFF, che disabilita, appunto, la visualizzazione automatica.
I campi del record sono in genere più di uno, così occorre indicare su quale effettuare la ricerca, nel nostro caso questa è sul campo PROFESSIONE. Siccome la ricerca deve interessare l'intero file, viene definito un ciclo DO WHILE...ENDDO, condizionato dal raggiungimento della fine del file (EOF).
Internamente al ciclo è imposta una condizione che determina la visualizzazione del nome (DISPLAY NOME) solo se il contenuto del campo SOCI è "RAGIONIERE". Si noti che la condizione IF va conclusa con ENDIF. L'ultima istruzione, SKIP, comunica semplicemente al "dBase II" di passare al successivo record.

Il Discovery 1

La seconda alternativa ai Microdrive della Sinclair

In precedenza si è descritto il dispositivo Wafadrive della Rotronics (vedere a pag. 1608), più maneggevole ed affidabile del Microdrive ma, a conti fatti, più lento e sempre basato sull'impiego di nastri magnetici ad anello.
Quello descritto qui, invece, è un apparecchio di tipo più tradizionale, a dischetti, prodotto dalla Opus Supplies (GB).
L'idea di un'unità a dischetti per lo Spectrum non è del tutto nuova, ma nessuno dei dispositivi immessi sul mercato ha avuto lo stesso successo del Discovery 1. Ciò è dovuto almeno in parte al fatto che spesso tali dispositivi sono reclamizzati in riviste molto specializzate facendo forse credere che siano diretti agli utenti più esperti. Inoltre, per nessuno di tali apparecchi è stata fatta una pubblicità su larga scala.
Il Discovery 1 è racchiuso in un contenitore metallico, dotato di una base piatta che funge da supporto per lo Spectrum, in modo da rendere più solido il collegamento al connettore d'espansione.
Tuttavia, anche se sembra un'operazione di facile esecuzione, l'innesto di un apparecchio nell'altro è reso difficoltoso dalla fastidiosa presenza dei cavetti per la TV e per il registratore.
L'inconveniente è più marcato nello Spectrum+ che non nello Spectrum in versione normale, per il quale il Discovery è stato originariamente progettato.
Siccome un collegamento instabile può arrecare gravi danni Spectrum, esso va attentamente controllato prima di passare all'accensione del sistema.
Una volta fissata correttamente, l'unità Di-

DISCOVERY 1

DIMENSIONI

mm. 300 x 210 x 75

INTERFACCE

Connettore "a pettine" di espansione, interfaccia parallela tipo Centronics, porta per joystick, uscita per video composito

FORMATO

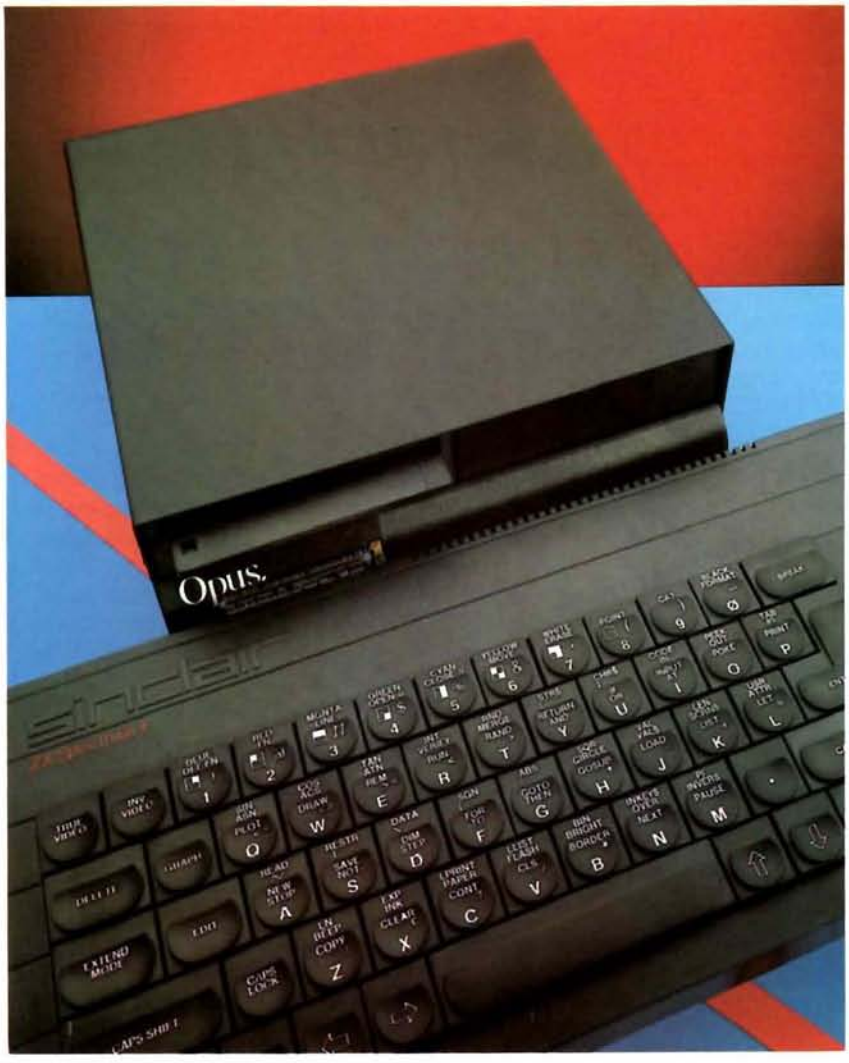
Dischetti tipo Sony da 3 e 1/2 pollici, a singola faccia, doppia densità

CAPACITÀ

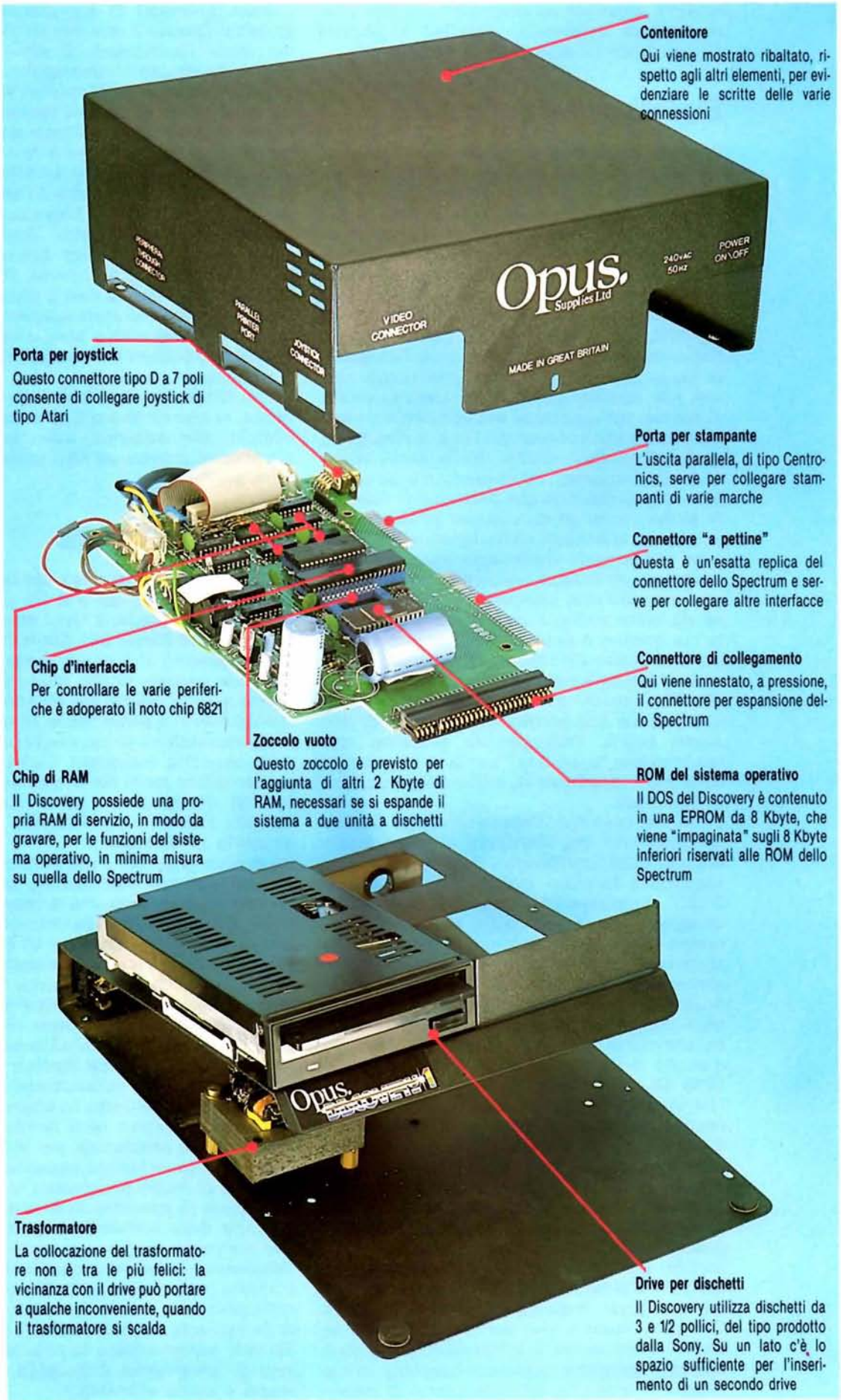
250 Kbyte totali, 180 Kbyte dopo la formattazione

VELOCITÀ

Trasferimento: 15 Kbaud
Accesso da traccia a traccia: 3 millisecondi



Alla scoperta di nuovi mondi
Il Discovery 1 intende essere un sistema di espansione "tutto in uno" per lo Spectrum. Oltre a possedere un'unità a dischetti tipo Sony (da 3 e 1/2 pollici) e relativo sistema operativo, il dispositivo offre il collegamento con joystick, con monitor a video composito e con altre periferiche, senza aggiunta di ulteriori interfacce



Contenitore

Qui viene mostrato ribaltato, rispetto agli altri elementi, per evidenziare le scritte delle varie connessioni

Porta per joystick

Questo connettore tipo D a 7 poli consente di collegare joystick di tipo Atari

Porta per stampante

L'uscita parallela, di tipo Centronics, serve per collegare stampanti di varie marche

Connettore "a pettine"

Questa è un'esatta replica del connettore dello Spectrum e serve per collegare altre interfacce

Chip d'interfaccia

Per controllare le varie periferiche è adoperato il noto chip 6821

Connettore di collegamento

Qui viene innestato, a pressione, il connettore per espansione dello Spectrum

Zoccolo vuoto

Questo zoccolo è previsto per l'aggiunta di altri 2 Kbyte di RAM, necessari se si espande il sistema a due unità a dischetti

Chip di RAM

Il Discovery possiede una propria RAM di servizio, in modo da gravare, per le funzioni del sistema operativo, in minima misura su quella dello Spectrum

ROM del sistema operativo

Il DOS del Discovery è contenuto in una EPROM da 8 Kbyte, che viene "impaginata" sugli 8 Kbyte inferiori riservati alle ROM dello Spectrum

Trasformatore

La collocazione del trasformatore non è tra le più felici: la vicinanza con il drive può portare a qualche inconveniente, quando il trasformatore si scalda

Drive per dischetti

Il Discovery utilizza dischetti da 3 e 1/2 pollici, del tipo prodotto dalla Sony. Su un lato c'è lo spazio sufficiente per l'inserimento di un secondo drive

scovery provvede all'alimentazione dell'intero sistema, rendendo superfluo il piccolo alimentatore in dotazione al computer.

Espansione a due dischetti

Sopra il connettore, a sinistra di chi guarda, c'è la fessura per i dischetti da 3 1/2 pollici, mentre a destra c'è spazio sufficiente per un secondo drive (una versione a doppia unità, chiamata Discovery 2, è in corso di produzione). Ciò fa ritenere che sia possibile aggiungere una seconda unità a dischetti anche in un momento successivo all'acquisto del Discovery 1. La Opus ha annunciato il lancio di un'unità dischetti da 5 e 1/4 pollici.
Seguendo la stessa tendenza della Rotronics, la Opus ha progettato un apparecchio che non solo fornisce una più efficiente memoria di massa, ma aggiunge anche interfacce per periferiche che consentono l'uso di stampanti o altri dispositivi esterni. Nella parte posteriore del Discovery, ad esempio, c'è un'uscita per video composito che permette di collegare lo Spectrum ad un monitor professionale (a quanto affermato dalla Opus, ciò è stato fatto per favorire chi intenda usare un word processor per lunghe sessioni di lavoro). È un vero peccato che, assieme a tale uscita, non ne sia stata prevista anche una di tipo RGB la cui qualità è notevolmente superiore.
Sul lato destro del Discovery, accanto al connettore bidirezionale per stampanti parallele tipo Centronics, c'è una porta per joystick (compatibile con joystick Kempston e di tipo Atari). Infine, l'apparecchio presenta una connessione "a pettine" per il collegamento delle altre interfacce di solito adoperate con lo Spectrum.
Come nel caso del Wafadrive, anche il sistema operativo del Discovery si adegua allo standard dell'Interfaccia 1. I comandi, infatti, seguono il formato <comando> *. Quando il BASIC raggiunge l'asterisco non riconosce il comando, ma il passaggio alla routine di errore viene intercettato dal software del Discovery, che "impagina" la propria ROM, contenente i comandi per dischetti, al posto degli 8 Kbyte inferiori di ROM dello Spectrum. Se il comando esiste esso viene eseguito, altrimenti il controllo passa alla routine d'errore della stessa ROM che contiene il DOS (Disk Operating System).
Nel progettare il DOS, la Opus ha ritenuto opportuno mantenere una piena compatibilità con i comandi per il Microdrive. Le ragioni di questa scelta sono molteplici: poiché lo Spectrum consente l'immissione di comandi in forma abbreviata (usando singoli tasti) è più coerente adeguarsi a questo criterio, piuttosto che creare comandi del tutto nuovi. Inoltre, poiché la sintassi è identica, chi è abituato a lavorare con il Microdrive passa a un immediato impiego dell'unità Discovery. Un terzo motivo, che da solo giustifica la scelta, è che meno si interviene sul sistema operativo meglio è: si evitano problemi con la mappa di memoria del computer e di conseguenza problemi di compatibilità tra programmi. Questo è uno dei motivi che hanno decretato l'insuccesso di altre periferiche, che in determinate circostanze, ad esempio, impedivano l'uso dell'Interfaccia 1.
La fedeltà con la quale i tecnici della Opus hanno seguito l'impostazione tracciata dalla Sinclair per il Microdrive è davvero ammirevole. Infatti, un comando del Microdrive quale LOAD * 'm';1;'nome' funziona in modo identico sul Discovery, solo che l'operazione avviene, ovviamente, su dischetto. Addirittura, nei comandi per il Discovery derivati da quelli per il Microdrive, la lettera 'm' può essere omessa, abbreviando così il comando. Anche altri comandi sono stati adattati: il carattere 't' (che col Microdrive apre il canale seriale RS232) qui consente l'accesso alla porta per la stampante parallela.
La suddivisione dei 16 canali di ingresso/uscita, numerati da 0 a 15, è identica: tre sono dedicati allo schermo, alla tastiera e alla stampante, mentre gli altri sono liberi per le altre periferiche.

I dischetti in azione

L'unità Discovery impiega dischetti a doppia densità tipo Sony da 3 e 1/2 pollici, ormai divenuti uno standard. Ogni dischetto contiene fino a 250 Kbyte (180 Kbyte liberi dopo la formattazione). Il sistema operativo prevede l'accesso casuale ai file, molto più rapido di quello seriale offerto da dispositivi similari. Inoltre non vi è alcun limite al numero di file memorizzabili su di un dischetto (all'interno della capacità massima), il che consente di gestire anche molti file di piccole dimensioni. In altri sistemi invece se si riempie l'indice dei file rimane una notevole quantità di memoria inutilizzata.
La ricerca di un file è più veloce sul Discovery che non sul Microdrive, ma l'effettivo caricamento in memoria è leggermente più lento. Ciò è dovuto alla minore velocità di trasferimento del Discovery (15 Kbaud contro i 19,2 del Microdrive). Il vantaggio più evidente del Discovery è la maggiore robustezza e compattezza dell'insieme, oltre all'uso di supporti magnetici di più ampia diffusione.
Scegliendo di adeguarsi all'impostazione dei comandi adottata dalla Sinclair per il Microdrive, la Opus ha evidentemente inteso prevenire, almeno in parte, la scarsità di software che caratterizza ogni novità del settore hardware. Di programmi per lo Spectrum ne esistono in grandissima quantità e, inoltre, lo sviluppo di nuovi programmi (o il passaggio di versioni in precedenza destinate ai nastri) da parte delle software house già affermate non dovrebbe presentare problemi.
Considerate le notevoli qualità del Discovery 1, si può ragionevolmente ritenere che il suo accoppiamento con lo Spectrum sia una formula vincente: molti possessori del computer Sinclair hanno adesso la possibilità di avvalersi di "vere" unità a dischetti, poco ingombranti e molto affidabili.

I/O sul Commodore 64

Le porte per le periferiche e le routine del SO di questo computer

Nell'illustrazione alla pagina seguente sono visibili tutte le porte del Commodore 64. Oltre a quella per il registratore, quella per i segnali audio/video e quella per il televisore, ci sono altre tre porte disponibili, da sinistra verso destra: la porta di espansione (usata per le cartucce), la porta seriale (o bus seriale) e la porta utente. Esaminiamole una alla volta.

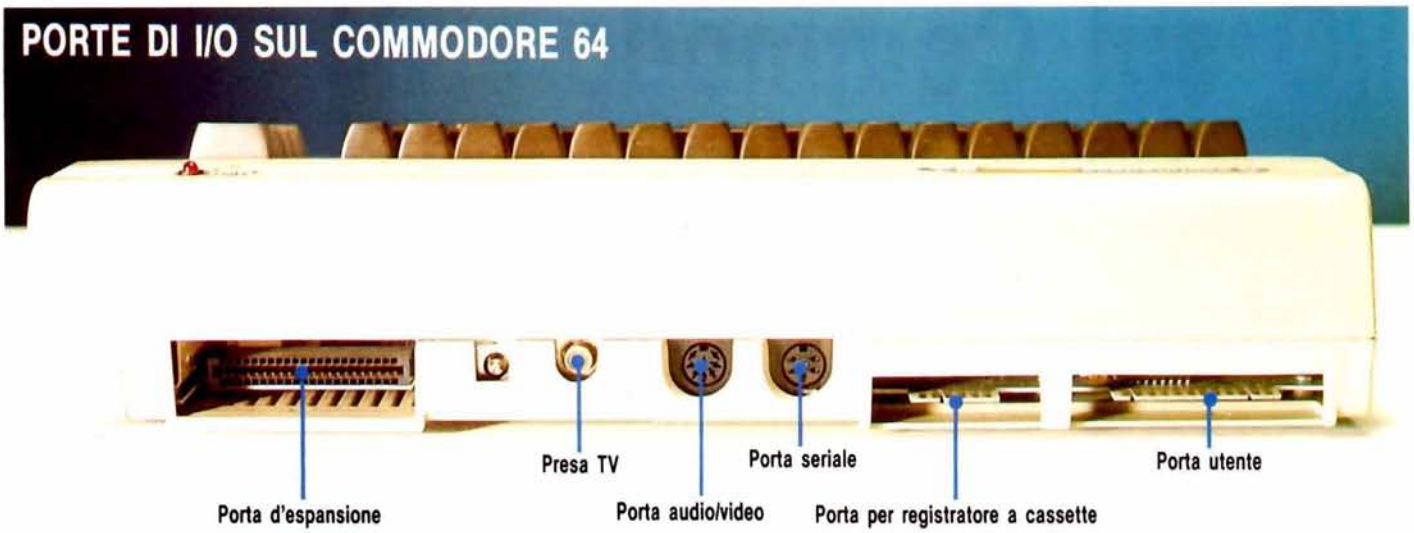
- *La porta seriale*: a questa presa (una DIN a sei poli) si possono collegare le stampanti seriali Commodore e le unità a dischetti 1541. L'istruzione OPEN si riferisce sempre a questa porta, salvo nel caso in cui si specifichi il numero di dispositivo 2 (questo corrisponde ad un'uscita RS232 sulla porta utente). Con il numero di dispositivo 8, ad esempio, il comando OPEN 2,8,2,"NOMEFILE", apre un file sull'unità a dischetti collegata alla porta seriale. Non è consigliabile tentare di adoperare questa porta seriale (né in BASIC né tramite il SO) per periferiche diverse da quelle della Commodore.
  Esistono in commercio alcune interfacce per convertire i segnali di questa porta da seriali a paralleli IEEE (spesso impiegate per utilizzare con il C64 periferiche, come l'unità a dischetti 4040, progettata per il PET).

- *La porta utente*: il connettore di questa porta, a 24 linee, è del tipo "a pettine". Attraverso di esso possono passare segnali sia paralleli che seriali. Questa porta può essere utilizzata, tra l'altro, per collegare stampanti che esulino dalla gamma Commodore (una Epson, ad esempio), trattandola come una periferica funzionante tramite RS232. Come il SO possa pilotare un dispositivo RS232 è spiegato più avanti.
  La porta utente può anche essere usata per comunicazioni parallele ad 8 bit, ma occorre scrivere (o reperire) apposite routine, poiché non sono presenti nel SO. Nella prossima parte di questa serie viene presentato un programma che mostra chiaramente l'impegno necessario per realizzare una simile routine di comunicazione parallela, basata sulle interrupt. Tale programma può essere utile per trasferire il contenuto di un blocco di memoria ad un dispositivo esterno (ad esempio un programma BASIC da un Commodore a un altro), purché entrambi gli apparecchi prevedano un livello dei segnali a 5 volt.

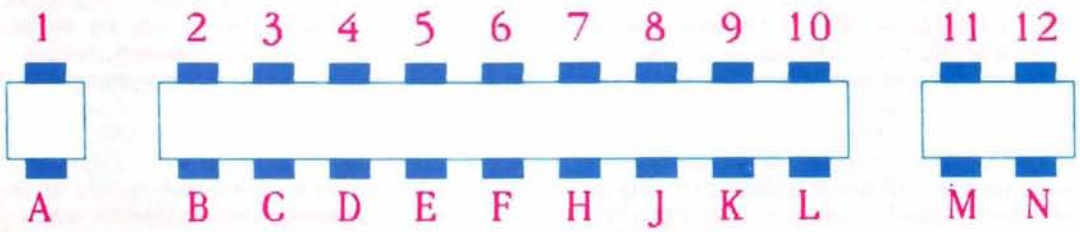
- *Porta di espansione*: Il connettore di questa porta (anch'esso a pettine) prevede 44 linee e consente di accedere a tutte le linee di controllo, al bus degli indirizzi e a quello dei dati del Commodore 64. Questa porta è utilizzata per interfacciare cartucce di giochi o cartucce con connessione parallela IEEE, adatte al collegamento con periferiche della gamma PET; è usata anche da dispositivi esterni, per acquisire un controllo quasi totale del computer.

Gli schemi di connessione per queste tre porte sono riportati nelle pagine seguenti.
Concentriamo l'attenzione, ora, su come il SO sfrutti tali porte di comunicazione. Il gruppo di routine del SO che gestice l'I/O (Input/Output), noto col nome "Kernel", sta alla base di tutti i comandi BASIC relativi all'ingresso/uscita di dati: OPEN, CLOSE, GET#, PRINT# e via dicendo. Fortunatamente, le routine del Kernel sono organizzate in modo da facilitarne l'accesso, in codice macchina, da parte del programmatore. In seguito è dato un breve programma in codice macchina, che consente di adoperare la routine di LOAD.
Inoltre, viene qui esaminata anche la possibilità del Commodore di comunicare tramite la porta RS232 incorporata (in particolare viene analizzato l'uso delle apposite routine per il collegamento di un modem).
Tuttavia, non è sempre possibile sfruttare le routine del SO per operazioni di I/O: talvolta occorre una comunicazione di tipo parallelo ad alta velocità, piuttosto che una seriale diretta ad un apparecchio distante. Purtroppo non esistono routine "già pronte", nel SO, che gestiscano la porta utente quindi non resta che realizzare le routine per comunicazioni parallele per conto proprio, programmando opportunamente i due CIA (Complex Interface Adaptor) 6526. Come già accennato, nella prossima parte di questa sezione viene fornito un programma di base per predisporre una comunicazione parallela. Tale routine consente di inviare o ricevere dati tramite la porta utente, anche durante l'esecuzione di un programma BASIC. Naturalmente, per ottenere un simile effetto (time sharing) è necessario ricorrere alle interrupt. Un esempio di programma-zeppa, basato su richieste d'interruzioni, si è già visto a pag. 1617: nel nostro caso, però, la routine è pilotata da "interrupt non mascherabili" (NMI), poiché la porta utente è in grado di generare segnali di questo tipo. La linea NMI è la seconda del processore 6510 e, diversamente dalla linea IRQ, non può essere "mascherata" usando i comandi SEI e CLI.

PORTE DI I/O SUL COMMODORE 64



LA PORTA UTENTE



Piedini superiori

Piedino	Sigla	Descrizione
1	GND	Massa
2	—	+5 volt, 100 mA massimi
3	—	RESET del sistema
4	CNT1	Contatore per la porta seriale CIA #1
5	SP1	Porta seriale CIA #1
6	CNT2	Contatore per la porta seriale CIA #2
7	SP2	Porta seriale CIA #2
8	PC2	Linea di "handshake" dal CIA #2
9	—	Linea collegata alla ATN della porta seriale
10	—	9 volt c.a. fase positiva
11	—	9 volt c.a. fase negativa
12	—	Massa

Piedini inferiori

Piedino	Sigla	Funzione RS232 (*)		(Vedere note)
A	GND	Massa dell'apparecchio		(1)
B	FLAG2	Dato ricevuto	· ingresso	(1)
C	PB0	Dato ricevuto	· ingresso	(1)
D	PB1	Richiesta d'invio (RTS)	· uscita	(2)
E	PB2	Terminale dati pronto (DTR)	· uscita	(2)
F	PB3	Indicatore di anello (RI)	· ingresso	-
H	PB4	Segnale linea ricezione (DCD)	· ingresso	(3)
J	PB5	(Non usato)	·	-
K	PB6	Abilitazione invio (CTS)	· ingresso	(3)
L	PB7	Gruppo dati pronto (DSR)	· ingresso	(3)
M	PA2	Dati trasmessi	· uscita	(2)
N	GND	Massa dei segnali		(2)

Note:

(*) Oltre ad essere indirizzabili dalle routine del Kernel, queste linee sono programmabili direttamente dall'utente per operazioni di I/O. Queste linee sono state usate nei vari progetti della serie "Laboratorio"

(1) Usata in entrambi i protocolli XON/XOFF e CTS/RTS

(2) Usata da entrambi i protocolli, ma a livello alto in XON/XOFF

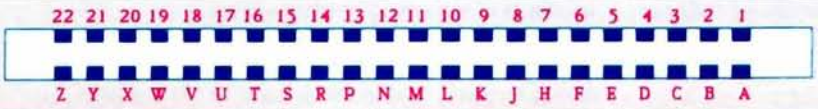
(3) Usata solamente nel protocollo CTS/RTS

LA PORTA SERIALE



Piedino	Sigla	Descrizione
1	SERIAL $\overline{\text{SRQIN}}$	Richiesta di attenzione
2	GND	Massa
3	SERIAL ATN I/O	Segnale di attenzione ai dispositivi
4	SERIAL CLK I/O	Temporizzatore della porta seriale
5	SERIAL DATA I/O	Linea di trasferimento di singoli bit (dati)
6	$\overline{\text{RESET}}$	Linea di reset dell'hardware

LA PORTA D'ESPANSIONE



Piedini superiori

Piedino	Sigla	Commenti
1	GND	Massa
2	+5v	
3	+5v	
4	$\overline{\text{IRQ}}$	Richiesta d'interrupt
5	R/$\overline{\text{W}}$	Abilitazione lettura/scrittura
6	Dot Clock	
7	I/O 1	
8	$\overline{\text{GAME}}$	
9	$\overline{\text{EXROM}}$	
10	I/O 2	
11	$\overline{\text{ROML}}$	
12	BA	
13	$\overline{\text{DMA}}$	
14	D7	
15	D6	
16	D5	
17	D4	8 linee dati
18	D3	
19	D2	
20	D1	
21	D0	
22	GND	Massa

Piedini inferiori

Piedino	Sigla	Commenti
A	GND	Massa
B	$\overline{\text{ROMH}}$	
C	$\overline{\text{RESET}}$	Linea di reset dell'hardware
D	$\overline{\text{NMI}}$	Interrupt non mascherabile
E	S 02	
F	A15	
H	A14	
J	A13	
K	A12	
L	A11	
M	A10	
N	A9	
P	A8	16 linee di indirizzamento
R	A7	
S	A6	
T	A5	
U	A4	
V	A3	
W	A2	
X	A1	
Y	A0	
Z	GND	Massa

IL KERNEL IN AZIONE

Il Kernel è una raccolta di routine di I/O, ognuna delle quali può essere richiamata da programmi scritti in BASIC o in codice macchina. L'area ROM occupata dal Kernel va da $E000 a $FFFF, ma le routine vengono chiamate tramite una "tabella di salti" (ossia una sequenza contigua di indirizzi) situata nella parte alta della memoria (vedere a pag. 1618). Il vantaggio di un accesso tramite tabella di salti è che, indipendentemente dalla versione del Kernel, l'indirizzo di chiamata delle routine non varia, mentre invece può cambiare il codice macchina di cui esse sono composte.

Per adoperare le routine del Kernel servono precise informazioni su ciascuna di esse (in particolare occorre sapere quali parametri sono necessari e in quali registri passarli): la migliore fonte è il manuale di consultazione per il programmatore, fornito dalla stessa Commodore. Ad esempio, per usare la routine del Kernel che opera la LOAD dei programmi, occorre un programma del tipo qui listato. Prima di adoperarlo, i caratteri che compongono il nome del file sul quale operare la LOAD vanno depositati, in sequenza, in qualche zona di memoria (qui individuata dall'indirizzo PHI/PLO):

```
LDA  #$01    Numero logico del file (lfn)
LDX  #$08    Numero del dispositivo (disco)
LDY  #$00    Indirizzo secondario (#$00 attiva la LOAD
             rilocabile)
JSR  $FFBA   Chiamata alla routine SLFS del Kernel (per
             fissare lfn e indirizzo secondario)
LDA  #$0A    Lunghezza del nome del file (qui 10)
LDX  PLO     Parte bassa dell'indirizzo per il nome del file
LDY  PHI     Parte alta dell'indirizzo per il nome del file
JSR  $FFBD   Chiamata alla routine SETNAM (attribuisci
             nome del file) del Kernel
LDA  #$00    Load = #$00 / Verify = #$01
LDX  DLO     Parte bassa dell'indirizzo di destinazione
LDY  DHI     Parte alta dell'indirizzo di destinazione
JSR  $FFD5   Chiamata alla routine LOAD del Kernel
```

LA RS232 SUL COMMODORE 64

Purché si operi con metodo, l'uso della RS232 tramite le routine del sistema operativo non presenta grossi problemi. Ci sono diverse considerazioni da tener presenti per il corretto impiego della RS232 sulla porta per l'utente, a parte quella di fissare la velocità di trasferimento in baud, esaminata in seguito:

- Il Commodore 64 funziona con livelli di segnale a 0 e +5 volt, mentre lo standard RS232 prevede i livelli −12 e +12 volt. Pertanto, salvo nei trasferimenti di dati tra apparecchi Commodore, occorre un convertitore per rendere tali livelli compatibili. Uno di questi è distribuito dalla Commodore, sotto forma di cartuccia.

- I codici ASC del Commodore differiscono dai normali codici ASCII, cosicché si rende necessario l'impiego di due tabelle di conversione (una per la trasmissione e una per la ricezione dei dati).

- Ogni volta che viene aperto (OPEN) un canale seriale RS232, il sistema operativo "scarica" automaticamente (CLR) il contenuto dei registri: il contenuto di tutte le variabili usate da un programma BASIC, pertanto, va perduto e le istruzioni GOSUB generano un messaggio d'errore quando si giunge alla rispettiva RETURN. Ciò avviene perché le routine del Kernel assegnano alla RS232 due buffer di 256 byte nella parte alta della memoria, senza tener conto della eventuale presenza di una parte di programma in tale zona, che viene quindi rovinata (per di più tale evento non viene segnalato).

- Se un programma BASIC è lungo oppure contiene molte assegnazioni di stringhe, prima o poi entra in funzione il cosiddetto "garbage collector", o "raccoglitore di rifiuti", che riorganizza l'utilizzazione della memoria. In tal caso parte dei dati ricevuti tramite la RS232 può andare perduta. È quindi consigliabile, durante l'impiego della porta, usare programmi BASIC di limitate dimensioni e senza troppe assegnazioni di variabili.

Per aprire un canale RS232 dal BASIC si usa il seguente formato:

```
OPEN 2,2,3,CHR$(CTRL) + CHR$(COM)
```

dove CTRL e COM sono, rispettivamente, il byte di controllo e di comando, che contengono le informazioni necessarie all'apertura del canale. Si noti che CTRL e COM devono essere due valori alfanumerici (oppure le PEEK di due locazioni precedentemente riempite) e non due variabili! Ciascun bit di questi due byte ha un preciso significato che viene riportato nelle tabelle. Per esempio, per aprire un canale RS232 con un bit di stop, una lunghezza dei dati di 7 bit, a 300 baud (byte CTRL = 38), parità pari, full duplex e flusso su tre linee (byte COM = 96), si adopera il comando:

```
OPEN 2,2,3,CHR$(38) + CHR$(96)
```

Prima di scegliere la velocità di trasferimento, occorre fare qualche considerazione. In trasmissione la velocità non è particolarmente critica, poiché in genere gli apparecchi riceventi possono tollerare piccole fluttuazioni (si può arrivare a 2400 baud per l'invio tramite RS232 in BASIC). Si tenga presente, comunque, che l'effettiva velocità di trasferimento è sempre leggermente inferiore ai baud selezionati, in parte a causa della presenza dei bit di start/stop che allungano la lunghezza totale dei dati.

La situazione, però, è alquanto diversa durante la ricezione di dati. In questo caso un programma BASIC riesce a malapena a estrarre un byte dal buffer d'ingresso e a visualizzarlo all'interno di un ciclo.

Per migliorare l'efficienza della ricezione, occorre in qualche modo fermare periodicamente il dispositivo che invia i dati, non appena il buffer d'ingresso si riempie. In genere la procedura, nel caso di un protocollo di ricezione a 3 linee è:

1) Leggere un modesto numero di byte (più sono i baud, minore deve essere questo numero), tramite una GET#2,A$, e utilizzarli immediatamente oppure depositarli in una matrice in attesa di un'ulteriore elaborazione.
2) Bloccare temporaneamente il dispositivo esterno di trasmissione, usando una PRINT#2,CHR$(17).
3) Leggere i byte rimanenti, fino a esaurimento, dal buffer d'ingresso della RS232 ed elaborare tutti i byte letti (controllando, nel frattempo, se vengono premuti tasti sulla tastiera, se si è raggiunta la fine del file (EOF), ecc.).
4) Riabilitare il dispositivo di trasmissione usando stavolta PRINT#2,CHR$(19) e ritornare al punto 1.

Una volta aperto un canale per la RS232, i byte vengono ricevuti o inviati ricorrendo alle semplici PRINT# e GET# (l'istruzione INPUT# va evitata). Il byte dello stato, ST, va periodicamente esaminato per verificare la presenza di errori, segnalati da ST = 0 o ST = 8. Infine, terminata l'operazione di ingresso/uscita, il canale va chiuso con una CLOSE. Attenzione perché anche questa istruzione, con la RS232, provoca automaticamente lo scarico dei registri.

Byte CTRL

Bit 7	6	5	4	3	2	1	0	Funzione
—	—	—	X	0	0	0	1	50 baud
—	—	—	X	0	0	1	0	75 baud
—	—	—	X	0	0	1	1	110 baud
—	—	—	X	0	1	0	0	134,5 baud
—	—	—	X	0	1	0	1	150 baud
—	—	—	X	0	1	1	0	300 baud
—	—	—	X	0	1	1	1	1200 baud
—	—	—	X	1	0	0	0	1800 baud
—	—	—	X	1	0	1	0	2400 baud
—	0	0	X	—	—	—	—	dati 8 bit
—	0	1	X	—	—	—	—	dati 7 bit
—	1	0	X	—	—	—	—	dati 6 bit
—	1	1	X	—	—	—	—	dati 5 bit
0	—	—	X	—	—	—	—	1 bit di stop
1	—	—	X	—	—	—	—	2 bit di stop

X = senza significato

Byte COM

Bit 7	6	5	4	3	2	1	0	Funzione
—	—	—	—	X	X	X	0	Protocollo a 3 linee
—	—	—	—	X	X	X	1	Protocollo X
—	—	—	0	X	X	X	—	Full duplex
—	—	—	1	X	X	X	—	Half duplex
—	—	0	—	X	X	X	—	Parità ignorata
0	0	1	—	X	X	X	—	Parità dispari
0	1	1	—	X	X	X	—	Parità pari
1	0	1	—	X	X	X	—	Invio Mark (par. disabilitata)
1	1	1	—	X	X	X	—	Invio Spazio (par. disabilitata)

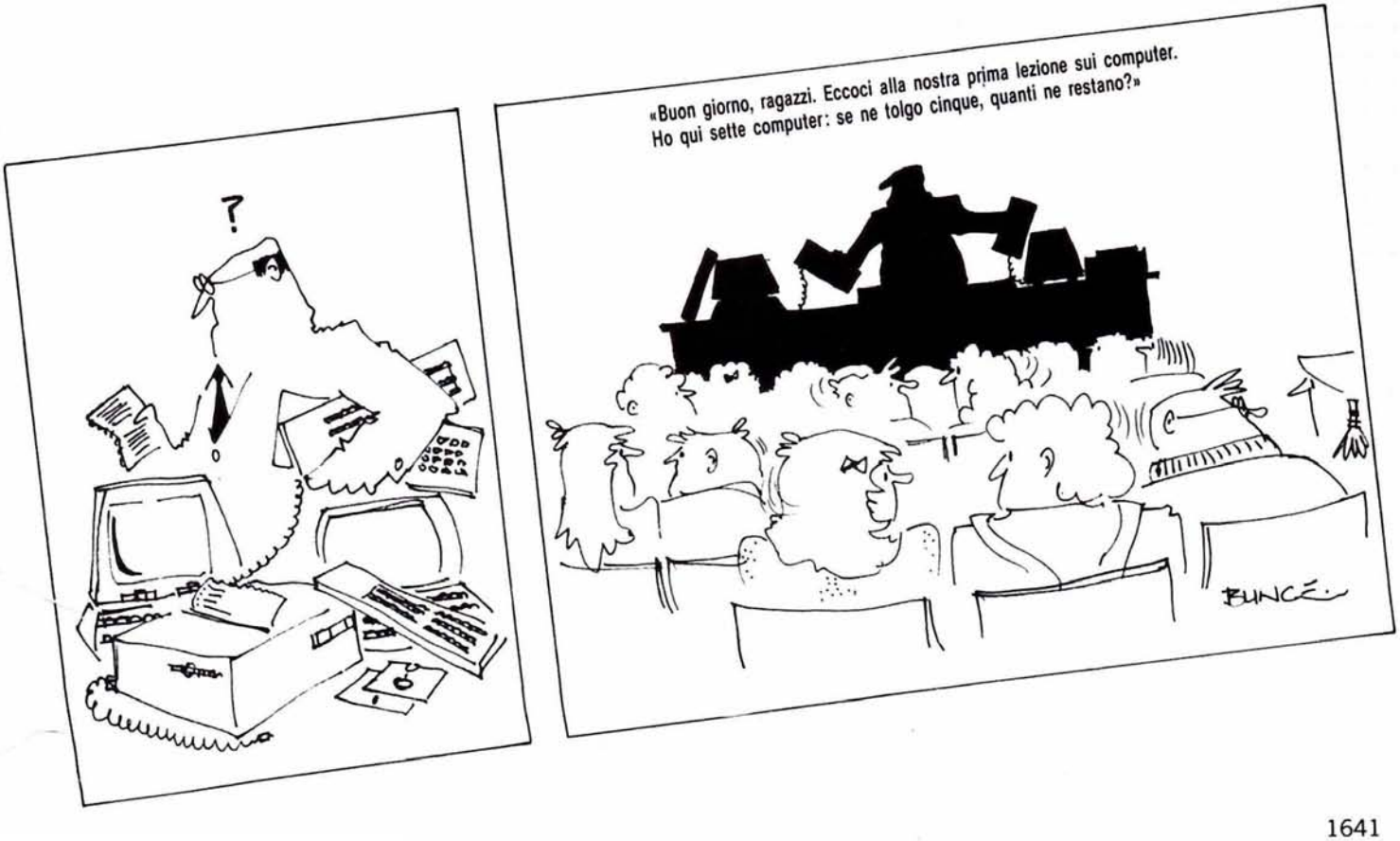
Una lezione di economia

Esperienze compiute all'estero insegnano che l'acquisto di hardware e di software deve essere oculato

L'introduzione del computer nell'ambito scolastico presenta molti problemi; la scarsa disponibilità di fondi, la notevole confusione che regna nel settore e le difficoltà nel formare insegnanti ben preparati non favoriscono certo uno sviluppo rapido ed efficace. La mancanza di fondi, ad esempio, può paralizzare l'attività didattica come nel caso in cui, una volta acquistato il computer, non si ha denaro sufficiente per il software.
Indubbiamente, per dar corso ad un progetto su larga scala, occorrono grossi finanziamenti. I progetti TICCIT e PLATO (negli USA) e il National Development Program in Gran Bretagna (vedere a pag. 1481), sono stati realizzati nel periodo pre-micro, ossia agli inizi degli anni '70, quando le disponibilità della finanza pubblica in quei due Paesi erano più sostanziose: in seguito le difficoltà economiche hanno causato notevoli tagli, frenando così gli entusiastici tentativi iniziali.

Una crisi delle strutture?

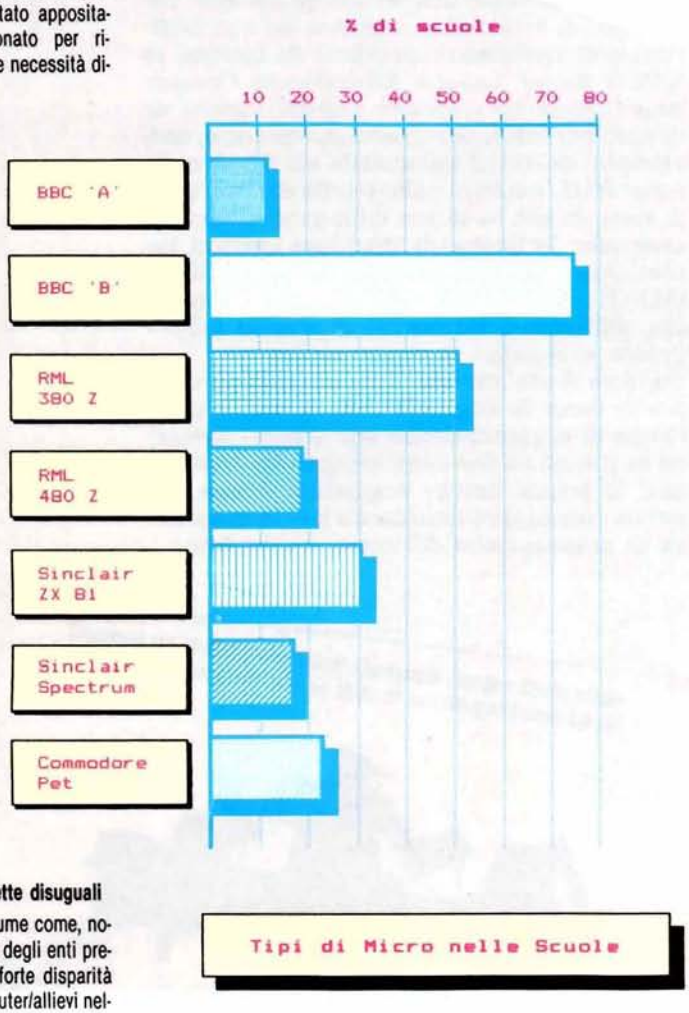
Oltre quello economico, esistono problemi di natura diversa. Addirittura, la stessa "collocazione" dei corsi di informatica non è ancora ben chiara: deve iniziare già alle elementari o essere circoscritta nell'ambito universitario o per lo meno nella scuola superiore?
Come sempre, dalle esperienze vissute (e soprattutto dagli errori commessi) in altre nazioni si possono trarre indicazioni utili. Un esempio di esperienza negativa (se non addirittura di inefficienza) proviene da Londra: lo ILECC (Inner London Educational Computing Centre) ha riversato ingenti somme di denaro per sviluppare, per conto proprio, una versione di LOGO utilizzabile sui micro computer RML, adottati nelle scuole del distretto. A metà strada la stessa ditta produttrice dei computer, la Research Machines Limited, ha realizzato un'ottima versione di LOGO e l'ILECC, di conseguenza, ha trovato non poche difficoltà a introdurre il proprio LOGO presso le scuole.
Un'altra fonte iniziale di inconvenienti, sempre in Gran Bretagna, è stata la totale mancanza di organizzazione per quanto concerne la preparazione degli insegnanti. In molti casi le scuole hanno acquistato anche più micro, rimasti poi inutilizzati per la mancanza di preparazione del corpo insegnante.



Le care bestiole

In queste pagine vengono riportati diagrammi relativi alla diffusione e all'utilizzo dei microcomputer nelle scuole in Gran Bretagna, una delle prime nazioni europee a gettarsi a capofitto in quella che può essere definita l'avventura informatica degli anni '80. Dai dati si può rilevare come l'apparecchio più diffuso sia il Micro BBC. Del resto questo computer è stato appositamente commissionato per rispondere a precise necessità didattiche

La situazione a livello universitario è risultata poco diversa: si sono tenuti nelle università alcuni corsi di aggiornamento per insegnanti, ma la loro durata (solo 20 ore) non bastava certo a fornire le basi necessarie.
Inoltre la carenza di software didattico di buona qualità, tuttora persistente, non è certo positiva. È difficile accertare le responsabilità di una tale situazione, che non può essere totalmente attribuita a mancanza di volontà da parte delle software house: alla conferenza annuale degli Editori Scolastici, nel 1983, è stato fatto presente che le case produttrici hanno investito grosse somme nello sviluppo di software destinato al settore didattico, ma che ciò ha causato perdite talmente ingenti da compromettere l'esistenza delle stesse dit-



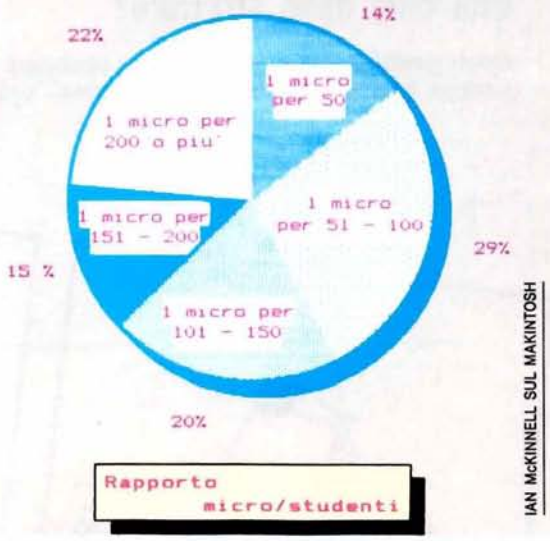
Una torta... con fette disuguali

Dal grafico si desume come, nonostante gli sforzi degli enti preposti, vi sia una forte disparità nel rapporto computer/allievi nelle scuole britanniche: più di una scuola su cinque può offrire un solo computer, per ben 200 studenti

te. Ciò è in parte dovuto ad alcune strutture pubbliche che, procurando gratuitamente alle scuole una certa quantità di software prodotto "artigianalmente", hanno impedito un vero e proprio sviluppo commerciale nel settore.
A titolo di esempio basta citare il caso del programma *Hopslide*, prodotto dalla Sloane Software. Si tratta di un programma molto semplice, indirizzato a bambini di età tra i 5 e i 7 anni, per l'apprendimento del valore e della collocazione naturale dei numeri. All'allievo viene presentata una fila di numeri (dal 5 al 9). Si può "saltare" da un numero all'altro scambiando le posizioni dei vari numeri fino ad ottenere la corretta sequenza. Il programma costa poche sterline ed era logico attendersi che avesse un discreto successo commerciale. Tuttavia, più o meno contemporaneamente, l'Inner London Education Authority (un ente statale che opera a Londra) tramite un suo centro di informatica, ha realizzato un programma pressoché identico (la serie di numeri è più ampia e la grafica è migliore) chiamato anch'esso *Hopslide*, inserendolo in un package di altri 19 programmi. Il tutto venduto a due sterline in più rispetto alla versione della Sloane.

Proteggere gli investimenti

Un aspetto pratico da non sottovalutare,

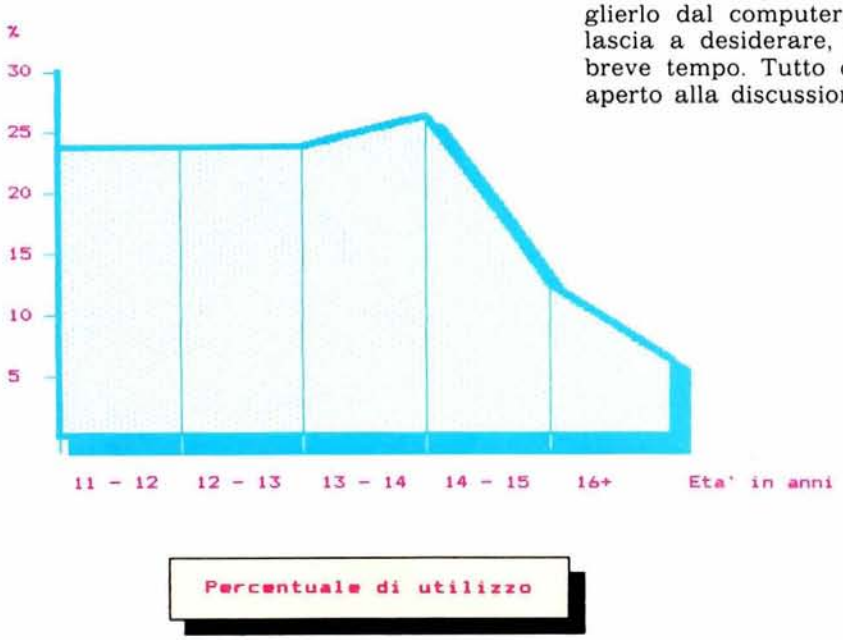


dell'uso dei computer in classe, è la protezione delle apparecchiature. Se, come è auspicabile, ogni classe verrà dotata di un proprio computer (non limitandosi, ad esempio, all'unico esemplare nell'aula di Fisica), si pone il problema di come evitarne il danneggiamento, anche involontario, da parte degli studenti o di malintenzionati. Non si tratta più di piccoli oggetti da laboratorio, facilmente riponibili in appositi ripostigli quando non vengono usati e non si può certo pretendere che il computer (con tutti gli annessi e connessi) venga rimosso ogni sera, per esser ricollocato in classe al mattino successivo.

Nelle esperienze fatte all'estero, dopo l'introduzione dei computer nella scuola, si è assistito perfino alla formazione di una nuova discriminazione di tipo sociale, in particolare tra maschi e femmine. Il computer, infatti, costituisce in genere una maggiore attrazione per la popolazione scolastica maschile. Del resto, gran parte degli home computer viene acquistata proprio da ragazzi, che tendono a sfogare su di essi, tramite giochi di tipo conflittuale, una parte della propria violenza. Ciò si riflette nell'ambiente scolastico, dove i (pochi) computer vengono spesso considerati dai maschi un appannaggio proprio. La situazione, ovviamente, può sfociare in un risultato del tutto negativo; limitare le conseguenze di questa tendenza spetta, almeno in parte, agli insegnanti.
Un'altra difficoltà che investe il settore è, spesso, l'errata prospettiva con la quale il problema viene affrontato dalle strutture pubbliche, che continuano a considerare l'insegnamento in termini di costi e produttività. Le conseguenze di una impostazione simile possono essere allucinanti: bambini inseriti in "catene di montaggio computerizzate" di una cultura tra virgolette, insegnanti sostituiti da disumani computer e via dicendo. Come lo studioso Seymour Papert ha fatto notare più volte, c'è una tendenza generale, nel mondo intero, a ridurre la società in confini sempre più rigidi e ad usare il computer in maniera tale da limitare fortemente i valori fondamentali della libertà umana. È possibile reagire contro una simile tendenza? Il modo esiste (e non è certo quello di gettare al rogo i computer): sforzarsi di adoperare questi apparecchi nella direzione opposta, moltiplicando le applicazioni che servono a migliorare, anziché a deteriorare, il tessuto sociale.
Il campo dell'insegnamento, in tutte le parti del mondo, sta senza dubbio attraversando un periodo molto critico ma, stranamente, ciò non è una conseguenza diretta dell'avvento dei computer. La necessità di rivedere, con occhio critico, i metodi ed i criteri adottati oggi nell'insegnamento prescinde dalla scelta di introdurre o meno i computer nelle classi. In una situazione così confusa, le opinioni sono molto contrastanti: c'è chi afferma l'assoluta necessità di usare i prodotti offerti dalla più recente tecnologia, c'è invece chi invita ad una pausa di riflessione, per non precipitare gli eventi e chi, infine, vorrebbe bandire del tutto i computer dalle scuole (almeno fino a un certo livello di istruzione), poiché ritiene che essi, in realtà, distraggano gli studenti dall'effettivo apprendimento delle materie.

Chi deve decidere?

I giudici, paradossalmente, potrebbero essere gli stessi bambini nelle scuole. Per meglio dire, essi, con il loro entusiasmo da una parte e lo spontaneo ma netto rifiuto dall'altra, possono fornire precise indicazioni circa la validità dei sistemi computerizzati di insegnamento. La qualità di un pezzo di software, infatti, la si può benissimo giudicare dalla reazione provocata negli studenti ai quali è destinato: se il programma è buono, lo studente apprende con grande entusiasmo anche le materie più ostiche ed è difficile distoglierlo dal computer. Se invece la qualità lascia a desiderare, l'interesse svanisce in breve tempo. Tutto ciò, ovviamente, lascia aperto alla discussione un altro aspetto del problema: se sia giusto o meno far svolgere questo duplice ruolo di cavie/giudici ai nostri figli e quale sia realmente il limite d'intervento giusto da applicare nella scelta del software o dell'uso stesso del computer.
Questi interrogativi rimarranno forse irrisolti per qualche anno ancora, finché, esaurita la corsa precipitosa causata da una nuova moda, si avrà più tempo per valutare nella giusta prospettiva gli effettivi pro e contro del computer in classe, si saranno sviluppati e diffusi linguaggi e programmi di più vasta portata e, infine, si sarà risolto il problema di un'adeguata formazione e preparazione degli insegnanti.



Chi manca all'appello?
Questo grafico illustra la percentuale di utilizzo di computer da parte di studenti nell'ambito scolastico, rispetto alla loro età. Si nota un forte decremento al crescere dell'età, dovuto principalmente alla scelta di una specializzazione che può non implicare l'uso di computer. È da notare che, tutto sommato, la percentuale di utilizzo non supera il 30%

Ottime relazioni

Come è organizzato l'accesso alle informazioni nei database manager

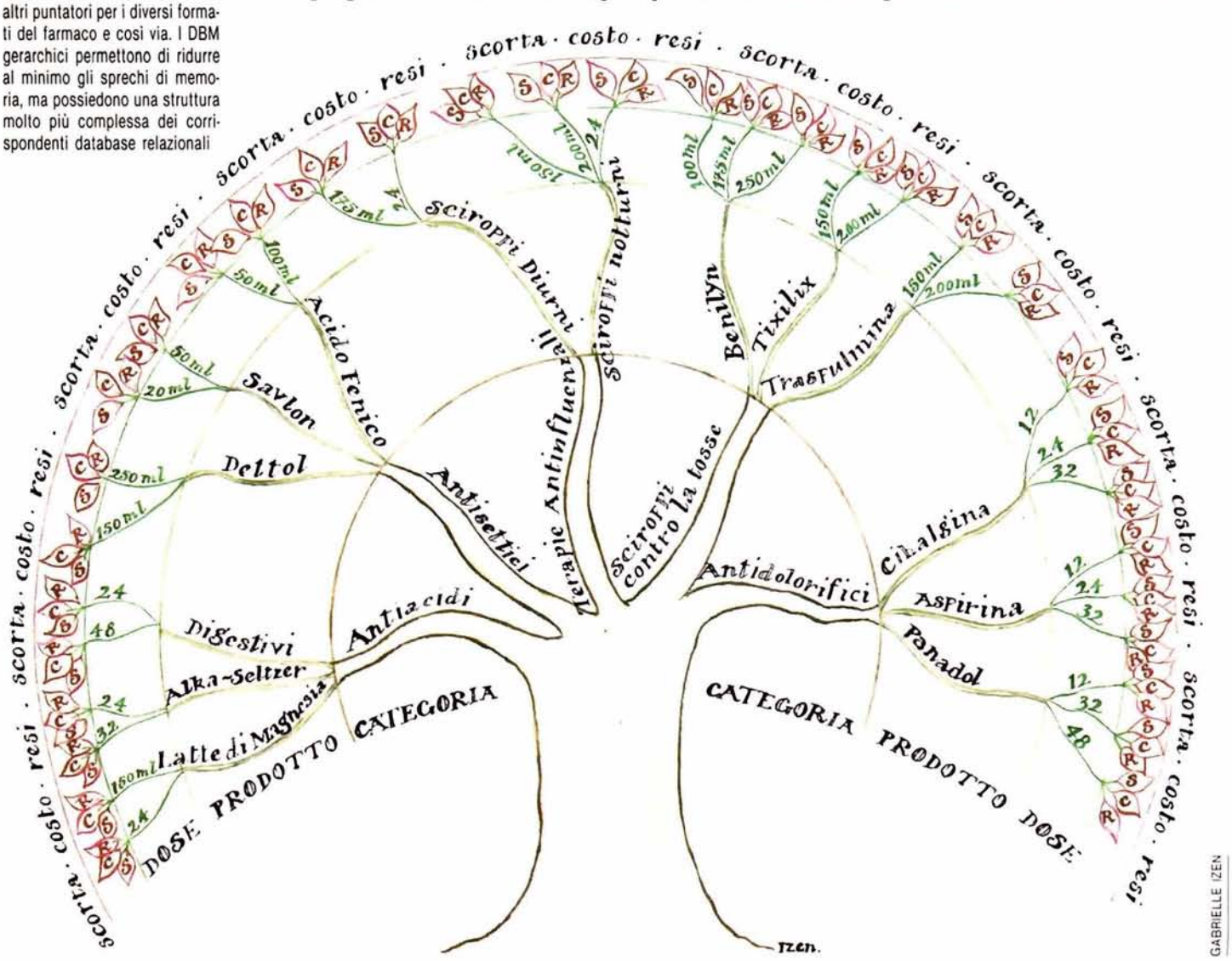
Molti DBM prodotti per gli home computer vengono detti sistemi "relazionali" per la gestione di archivi. Nei casi più semplici, ciò significa che ogni record di un file DBM ha la struttura di una tabella, composta da righe e colonne, simile a una lavagna elettronica. Le diverse tecniche adottate per presentare le informazioni non devono trarre in inganno: la struttura di base è sempre quella di una tabella o di una griglia. Inoltre l'organizzazione interna dei database è simile a quella dei comuni schedari: tutti i record possiedono un'identica struttura e i campi che li compongono hanno lunghezza fissa.

Il termine "relazionale" significa che le righe del database sono rigorosamente connesse alle colonne. Questo tipo di organizzazione manca di flessibilità, ma favorisce la creazione di programmi abbastanza semplici per la gestione dell'archivio. Poiché gli home computer, compresi gli apparecchi a 16 bit dotati di 128 Kbyte o più di memoria, possiedono una memoria abbastanza limitata, una ridotta velocità d'elaborazione e una limitata capacità di immagazzinamento dei dati, solo con alcune restrizioni ai pacchetti per la gestione dei database si riesce ad ottenere una potenza di elaborazione accettabile.

Un metodo di strutturazione delle informazioni, alternativo a quello "tabellare", è l'ordinamento "gerarchico". Questa tecnica consente di organizzare i dati secondo una struttura ad albero, in cui ogni ramo si suddivide in rami minori da cui originano ramoscelli e così via, fino ad arrivare alle foglie. Per apprezzare la differenza tra i due sistemi, qui di seguito viene descritta l'impostazione di un database per una rivendita di riviste e

L'albero... della salute

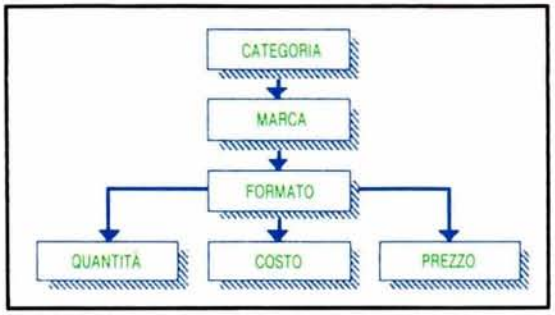
Mentre un DBM relazionale immagazzina le informazioni all'interno di tabelle, un DBM gerarchico organizza i propri dati in forma di "albero". Nell'esempio in figura, ogni categoria di medicinale contiene dei puntatori per i diversi nomi di farmaci disponibili, ciascuno dei quali possiede altri puntatori per i diversi formati del farmaco e così via. I DBM gerarchici permettono di ridurre al minimo gli sprechi di memoria, ma possiedono una struttura molto più complessa dei corrispondenti database relazionali



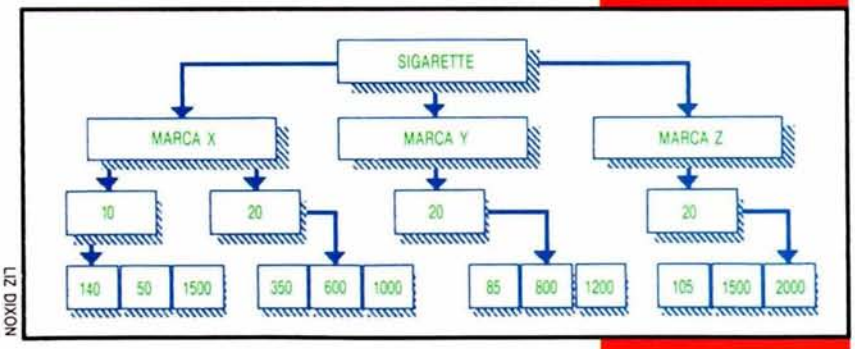
tabacchi, prima in forma relazionale quindi secondo un ordinamento gerarchico.

CATEGORIA	MARCA	FORMATO	SCORTE	COSTO	PREZZO
SIGARETTE	EMBASSY	20	350	1800	2000
SIGARETTE	EMBASSY	10	140	910	1000
SIGARETTE	DUNHILL INT	20	85	1790	2000
SIGARETTE	SILK CUT	20	105	1805	2000
DOLCIUMI	BOUNTY	1	106	800	1100
DOLCIUMI	POLO	1	95	300	550
DOLCIUMI	PIP	1	25	260	400
RIVISTA	I GIORNI	1	35	800	1000
RIVISTA	QUESTO MESE	1	12	1100	1800
RIVISTA	ECHI NAZIONALI	1	86	3800	4500
BEVANDE	COLA GOLD	33	35	900	1200
BEVANDE	ARANCIATA	33	40	880	1200
BEVANDE	BIRRA CERVO	33	35	1160	1400

Usando la prima tecnica, il database potrebbe avere la seguente struttura:



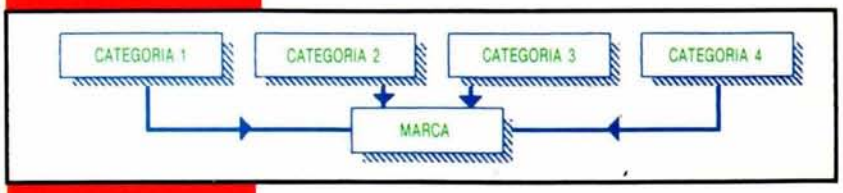
I record non sono suddivisi in campi, bensì in segmenti di dati, ciascuno dei quali può contenere più di un campo. Le registrazioni relative alle SIGARETTE, organizzate nel database in modo gerarchico, presentano invece la seguente struttura:



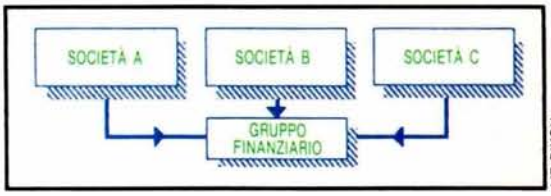
LIZ DIXON

Come mostra il diagramma, ogni elemento dell'informazione è superiore o sottoposto ad un altro dato e, per passare da un segmento di informazione a un altro, occorrono appositi puntatori. Nell'esempio precedente, la categoria SIGARETTE si serve di alcuni puntatori per accedere alle marche di sigarette e di altri per i formati, le quantità, i prezzi e così via. Supponiamo che la rivendita possieda un centinaio di categorie di magazzino, ma che gli articoli in vendita siano un migliaio. Con un database relazionale sono necessari 1.000 record, uno per ogni prodotto. Al contrario, un database gerarchico usa solo 100 segmenti per l'intera serie di articoli. Ovviamente la sua struttura è molto più complessa.

Esiste tuttavia un terzo tipo di organizzazione dei database, detto CODASYL o sistema "a rete", le cui caratteristiche standard sono state definite dal gruppo di lavoro sui database della Conferenza sui Linguaggi dei Sistemi per i Dati. Nei database gerarchici, il flusso dei dati può essere solo unidirezionale: ciò significa che una diramazione della struttura ad albero non può originare contemporaneamente da due rami. Tuttavia, nel database descritto in precedenza, una stessa marca può essere una diramazione di due o più categorie, ad esempio sigari e sigarette. In questo caso, le relazioni che intercorrono tra gli articoli in vendita e le categorie di magazzino si complicano notevolmente.
I sistemi CODASYL si basano su una rete di insiemi, ciascuno dei quali è composto da una serie di record. Questi, diversamente dai record dei database relazionali, non hanno una lunghezza fissa e possono fare parte di più insiemi contemporaneamente. Quindi, nei database CODASYL, un insieme può essere composto da un solo record, ma un record non può appartenere a più di un elemento dello stesso tipo d'insieme. Non sono dunque consentite strutture del tipo:



Questa limitazione crea dei problemi. Nel database della rivendita di giornali, bibite e tabacchi, difficilmente si avranno categorie come SIGARETTE, GIORNALI e BIBITE tutte riferite ad una stessa marca ma, in quello per agenti di borsa, ad esempio, un gruppo finanziario può essere azionista di più società:



LIZ DIXON

I database gerarchici e CODASYL sono più flessibili di quelli relazionali, ma la loro struttura è molto più complessa e richiede una potenza d'elaborazione che solo i minicomputer o i "mainframe" possono offrire. Gli home computer usano invariabilmente database di tipo relazionale. La difficoltà più comune che si incontra con questi sistemi riguarda la creazione di una lista di articoli da acquistare o catalogare (ad esempio, libri o altre merci) ciascuno dei quali viene commercializzato da più di un fornitore. In questo caso è necessario creare un record per gli articoli e una serie di record collegati per i diversi fornitori di uno stesso articolo. I DBM multifile permettono di aprire più file contemporaneamente e di consultare un file ausiliario pur trovandosi all'interno di un altro file.

Linee di comunicazione

L'invio o la ricezione di un blocco di memoria tramite la porta utente

Il Commodore 64 possiede due chip CIA 6526 (Complex Interface Adaptor), dedicati al controllo delle comunicazioni con l'esterno, oltre ad altri chip che gestiscono aspetti specifici del sistema di I/O: il 6510 e quelli del video. Un chip 6526 prevede due porte a 8 bit per i dati, entrambe dotate di linee programmabili individualmente. Inoltre, può gestire comunicazioni a 8 e 16 bit e incorpora 2 temporizzatori a 16 bit collegabili tra loro. Infine dispone di un registro di scorrimento a 8 bit per comunicazioni seriali e di un orologio "giornaliero" programmabile su 24 ore (vedere a pag. 1619).
Questo chip possiede anche due linee di sincronizzazione con funzioni particolari: PC e Flag. La linea PC viene posta a livello basso (0 logico), per la durata di un ciclo, dopo il trasferimento di un dato sulla porta B del 6526 e può essere usata per segnalare ad una periferica la condizione di disponibilità del dato. La linea Flag viene spesso utilizzata come segnale di controllo in ingresso, proveniente da un altro apparecchio. Questo segnale può essere usato per selezionare il bit Flag del registro delle interrupt che, in questa configurazione, provoca la generazione di un'interrupt NMI per il processore 6510.
Sul Commodore 64, i due chip 6526 svolgono differenti funzioni di I/O: il CIA #1 (con indirizzo di base $DC00) controlla la tastiera e i joystick mentre il CIA #2 (con indirizzo di base $DD00) gestisce i dati sulla porta seriale e su quella per utente. Inoltre il chip per lo schermo controlla l'I/O del monitor e il 6510 la porta per il registratore a cassette.
Il programma *Cuneopar* illustra in modo esauriente le procedure necessarie per programmare direttamente il 6526 per operazioni di ingresso/uscita. La routine è un cuneo NMI che sfrutta la linea Flag. La sua funzione è quella di trasmettere un particolare blocco di memoria attraverso la porta utente sotto forma di dati paralleli o, alternativamente, di ricevere dati paralleli a 8 bit finché il blocco di memoria che li immagazzina non è pieno. Poiché la routine funziona come un cuneo (vedere a pag. 1619), essa sincronizza la ricezione o la trasmissione dei dati in base alle interrupt NMI e contemporaneamente lascia libero il computer di svolgere altri compiti. Tuttavia, una velocità di trasmissione dei dati troppo elevata può costringere il Commodore 64 a dedicare tutto il proprio tempo all'esecuzione delle routine di servizio NMI, "congelando" le altre operazioni.
Il programma *Cuneopar* predispone due linee parallele a 8 bit per la comunicazione, tramite la porta utente, con un apparecchio esterno, ad esempio un secondo computer o una stampante parallela. I piedini da PB0 a PB7 di questa porta vengono usati per il trasferimento dei dati. Il piedino Flag 2 è l'ingresso di sincronizzazione del sistema di comunicazione e PA2 e PC2 segnalano rispettivamente le condizioni di "pronto per il dato" e "dato valido". Prima di lanciare il programma è necessario specificare l'area di RAM dalla quale verranno estratti i dati in uscita o ricevuti i dati in ingresso. Gli indirizzi iniziale e finale di quest'area vengono comunicati al programma (in forma byte basso/byte alto) inserendoli nelle locazioni da 50768 a 50771. La locazione 50772 viene usata per specificare se il programma trasmette o riceve i dati. Nel primo caso la locazione deve contenere il valore 1, nel secondo, il valore 0. Una volta definiti questi parametri, l'utente può lanciare la routine digitando il comando SYS 50775. Poiché il programma è pilotato da interrupt, le operazioni di trasmissione o ricezione dei dati avvengono "in sottofondo", ossia contemporaneamente all'introduzione o all'esecuzione di un altro programma BASIC.

PROGRAMMA "CUNEOPAR" DEL COMMODORE 64

Caricatore BASIC

```
1000 REM ** CARICATORE BASIC DI CUNEOPAR **
1010 DATA173,84,198,208,61,169,0,141,3
1020 DATA221,169,144,141,13,221,173,2
1030 DATA221,9,4,141,2,221,173,0,221,9
1040 DATA4,141,0,221,173,80,198,133,251
1050 DATA173,81,198,133,252,173,24,3
1060 DATA141,85,198,173,25,3,141,86,198
1070 DATA120,169,188,141,24,3,169,198
1080 DATA141,25,3,88,96,169,255,141,3
1090 DATA221,169,144,141,13,221,173,24
1100 DATA3,141,85,198,173,25,3,141,86
1110 DATA198,120,169,234,141,24,3,169
1120 DATA198,141,25,3,88,96,169,144,44
1130 DATA13,221,240,36,173,1,221,145
1140 DATA251,230,251,208,2,230,252,173
1150 DATA82,198,197,251,173,83,198,229
1160 DATA252,144,49,173,0,221,41,252
1170 DATA141,0,221,9,4,141,0,221,108,85
1180 DATA198,169,144,44,13,221,240,246
1190 DATA177,251,141,1,221,230,251,208
1200 DATA2,230,252,173,82,198,197,251
1210 DATA173,83,198,229,252,144,3,108
1220 DATA85,198,120,173,85,198,141,24,3
1230 DATA173,86,198,141,25,3,88,108,24
1240 DATA3
1250 DATA25596:REM*CHECKSUM*
1260 CC=0
1270 FOR I=50775TO50971
1280 READX:CC=CC+X:POKEI,X
1290 NEXT
1300 READX:IFCC<>XTHENPRINT"CHECKSUM ERROR"
1310 END
```

Caricamento di "Cuneopar"

Il listato in Assembly del programma "Cuneopar"(allelo) va caricato in memoria e tradotto in codice macchina con un assembler. In alternativa, il programma può essere trascritto sotto forma di istruzioni DATA, per poi lanciare il programma di caricamento in BASIC

Listato Assembly

```
;*************************************************
;*                                               *
;* CUNEOPAR - PROGRAMMA DI TRASMISSIONE/RICEZIONE *
;*   PER COMUNICAZIONI PARALLELE A 8 BIT         *
;*            SUL COMMODORE 64                   *
;*                                               *
;*************************************************
;
    CIA2 = $0000        ;INDIRIZZO DI BASE DEL CHIP 6526
    USCITA = $FF
    INGRESSO =$00
    OUTSHK = $04
    INTMSK = $90
    TOGHI = $04
    TOGLO = $FC
    NMIVETT = $0318
    ZPTEMP = $FB
    * = $C650
    INIZIO *=*+2        ;INDIRIZZO INIZIALE
    FINE *=*+2          ;INDIRIZZO FINALE
    MODO *=*+1          ;FLAG DI INGRESSO/USCITA
    VETTORE *=*+2       ;MEMORIA PER IL VETTORE NMI

    LDA MODO            ;INGRESSO O USCITA
    BNE OUTDAT          ;SALTA SE INGRESSO
    LDA #INGRESSO
    STA CIA2+3          ;SELEZIONA IL DDR (REG. DIREZIONE DATI)
                         PER L'INGRESSO
    LDA INTMSK
    STA CIA2+13         ;DISABILITA LE INTERRUPT DI FLAG
    LDA CIA2+3
    ORA #OUTSHK
    STA CIA2+2          ;SELEZIONA PA2 PER L'USCITA
    LDA CIA2
    ORA TOGHI
    STA CIA2            ;PONE ALTA LA LINEA DI "HANDSHAKE" PA2
    LDA INIZIO
    STA ZPTEMP
    LDA INIZIO+1        ;SPOSTA PUNTATORI IN PAGINA 0
    STA ZPTEMP+1
   }
   ;INIZIALIZZA IL CUNEO D'INGRESSO
   }
    LDA NMIVETT
    STA VETTORE         ;SALVA IL VECCHIO VETTORE NMI
    LDA NMIVETT+1
    STA VETTORE+1
    SEI                 }
    LDA #<NXTIN         }
    STA NMIVETT         ;INSERISCE IL CUNEO DI INGRESSO-DATI
    LDA #>NXTIN         }
    STA NMIVETT+1       }
    CLI
    RTS
   }
   ;INIZIALIZZA IL CUNEO D'USCITA
   }
OUTDAT
    LDA #USCITA
    STA CIA2+3          ;SELEZIONA IL DDR PER L'USCITA
    LDA #INTMSK
    STA CIA2+13         ;DISABILITA LE INTERRUPT DI FLAG
   }
   }
    LDA NMIVETT
    STA VETTORE
    LDA NMIVETT+1       ;SALVA IL VECCHIO VETTORE NMI
    STA VETTORE+1
    SEI                 }
    LDA #<NXTOUT        }
    STA NMI VETT        ;INSERISCE IL CUNEO DI USCITA-DATI
    LDA #>NXTOUT        }
    STA NMIVETT+1
    CLI                 }
    RTS
   }
   }
   ;ROUTINE DI SERVIZIO PER L'INGRESSO DI DATI
   }
   }
NXTIN
    LDA #INTMSK         ;VERIFICA ICR
    BIT CIA2+13         ;INTERRUPT CAUSATA DA FLAG?
    BEQ NOTCOM          ;NO ... UNA COMUNE NMI
   }
   ;BYTE SULLA PORTA
   }
    LDA CIA2+1          ;LEGGE UN BYTE
    STA (ZPTEMP),Y      ;LO DEPOSITA IN MEMORIA
    INC ZPTEMP
    BNE TEST1           ;INCREMENTA IL PUNTATORE
    INC ZPTEMP+1
   }
TEST1
    LDA END
    CMP ZPTEMP
    LDA END+1           ;VERIFICA SE FINITO
    SBC ZPTEMP+1
    BCC DONE            ;SE FINITO, SALTA
   }
   ;COMUNICA DI ESSERE PRONTO PER BYTE SUCC.
   }
    LDA CIA2
    AND #TOGLO
    STA CIA2            ;PONE PA2 PRIMA BASSA, POI ALTA
    ORA #TOGHI
    STA CIA2
   }
   ;ESEGUE LA NORMALE ROUTINE NMI
   }

NOCOM
    JMP (VETTORE)


   }
   }
   ;ROUTINE DI SERVIZIO PER L'USCITA DI DATI
   }
   }
NXTOUT
    LDA #INT MSK        ;VERIFICA ICR
    BIT CIA2+13         ;INTERRUPT  CAUSATA DAL FLAG?
    BEQ NOTCOM          ;NO ... UNA COMUNE NMI
   }
   ;BYTE TRASMESSO
   }
    LDA (ZPTEMP),Y      ;ESTRAE UN BYTE DALLA MEMORI
    STA CIA2+1          ;LO TRASMETTE. (PC PASSA A LIVELLO
                        ;BASSO PER 1 CICLO)
    INC ZPTEMP
    BNE TEST2           ;INCREMENTA IL PUNTATORE
    INC ZPTEMP+1

TEST2
    LDA END
    CMP ZPTEMP
    LDA END+1           ;VERIFICA SE FINITO
    SBC ZPTEMP+1
    BCC DONE            ;SE FINITO, SALTA
   }
   ;ESEGUE LA NORMALE ROUTINE NMI
   }
    JMP (VETTORE)
   }
   }
   ;FINITA RIMOZIONE CUNEO
   }
   }
    DONE
    SEI
    LDA VETTORE
    STA NMIVETT         ;RIPRISTINA IL VALORE ORIGINALE
    LDA VETTORE+1       ;NEL VETTORE NMI
    STA NMIVETT+1
    CLI
    JMP (NMIVETT)
```

Il programma, riprodotto per gentile concessione degli Autori e della Ellis Horwood Ltd., è tratto da: "Mastering the Commodore 64" di Jones & Carpenter

DATABASE

La terza tabella della mappa di memoria del Commodore 64. Per concessione della Commodore Business Machines

INDIRIZZO ESADECIMALE	ETICHETTA	LOCAZIONE DECIMALE	DESCRIZIONE
0047·0048	VARPNT	71·72	**Pointer: Current BASIC Variable Data** Puntatore ai dati variabili correnti del BASIC
0049·004A	FORPNT	73·74	**Pointer: Index Variable for FOR/NEXT** Puntatore alla variabile "indice" per cicli FOR/NEXT
004B·0060		75·96	**Temp Pointer / Data Area** Puntatore temporaneo / Area Data
0061	FACEXP	97	**Floating·Point Accumulator #1: Exponent** Accumulatore #1 per virgola mobile: Esponente
0062·0065	FACHO	98·101	**Floating Accum. #1: Mantissa** Accumulatore #1 per virgola mobile: Mantissa
0066	FACSGN	102	**Floating Accum. #1: Sign** Accumulatore #1 per virgola mobile: Segno
0067	SGNFLG	103	**Pointer: Series Evaluation Constant** Puntatore: Costante per valutazione di serie
0068	BITS	104	**Floating Accum. #1: Overflow Digit** Accumulatore #1 per virgola mobile: cifra in eccesso
0069	ARGEXP	105	**Floating·Point Accum. #2: Exponent** Accumulatore #2 per virgola mobile: Esponente
006A·006D	ARGHO	106·109	**Floating Accum. #2: Mantissa** Accumulatore #2 per virgola mobile: Mantissa
006E	ARGSGN	110	**Floating Accum. #2: Sign** Accumulatore #2 per virgola mobile: Segno
006F	ARISGN	111	**Sign Comparison Result: Accum. #1 vs #2** Segno risultante dal confronto tra Accumulatore #1 e Accumulatore #2
0070	FACOV	112	**Floating Accum. #1: Low·Order (Rounding)** Accumulatore #1 per virgola mobile: Arrotondamento
0071·0072	FBUFPT	113·114	**Pointer: Cassette Buffer** Puntatore al buffer per le cassette
0073·008A	CHRGET	115·138	**Subroutine: Get Next Byte of BASIC Text** Subroutine: Estrae il prossimo byte di testo BASIC
0079	CHRGOT	121	**Entry to Get Same Byte of Text Again** Punto d'ingresso per estrarre nuovamente lo stesso byte di testo
007A·007B	TXTPRT	122·123	**Pointer: Current Byte of Basic Text** Puntatore al corrente byte di testo BASIC
008B·008F	RNDX	139·143	**Floating RND Function Seed Value** "Seme" per la funzione RND a virgola mobile
0090	STATUS	144	**Kernal I/O Status Word: ST** Parola di stato per I/O del Kernal: ST

Il Penman Plotter

Una periferica polifunzionale per qualsiasi computer dotato di interfaccia RS232

La vasta dotazione di interfacce del Micro BBC ha favorito l'impiego di questo apparecchio in campo educativo e ha incoraggiato la produzione di numerose periferiche di tipo didattico. Tra queste vanno ricordati plotter, mouse e vari robot da pavimento. Tuttavia l'unità più interessante è probabilmente il Penman Plotter, che può svolgere le funzioni di tre diverse periferiche.

Il Penman Plotter, corredato di apposito software, è composto da un'unità di controllo, un gruppo di tracciamento mobile, un sistema di alimentazione e un cavo di collegamento RS232. Quando non è in funzione, la parte mobile del plotter viene fissata all'unità di controllo formando un unico corpo di ridotte dimensioni (335 x 128 x 55 mm.). Per estrarlo basta premere un pulsante di bloccaggio posto sotto la base e svolgere il cavo a nastro di collegamento, alloggiato nell'unità di controllo. Sul frontale del plotter sono visibili tre fessure, nelle quali vengono inserite le penne. Una quarta apertura, posta al centro del plotter, accoglie la penna usata per la grafica della tartaruga. Le penne sono in realtà piccoli pennarelli lunghi 40 mm. tenuti sospesi sul foglio da fermagli a molla. Quando un pennarello riceve un comando di tracciamento, la corrispondente molla viene attirata verso il basso da una elettrocalamita collegata e la punta del pennarello entra in contatto col foglio.

Il plotter si muove su tre ruote: due ne dirigono i movimenti mentre la terza, una piccola ruota di plastica a orientazione libera, è necessaria per l'equilibrio dell'unità. Le ruote "pilota" sono di metallo e possiedono un battistrada ruvido per migliorare l'aderenza con il foglio durante i movimenti del plotter. Due fotosensori, posti in vicinanza alle ruote, individuano i margini del foglio registrando la differenza di luminosità tra la carta e il piano di lavoro.

Le tre elettrocalamite per le penne sono collegate a piccole leve, interne al plotter, che azionano meccanicamente i pennarelli rispetto alla carta. Posteriormente ad esse ci sono due comuni motori elettrici collegati alle ruote motrici.

I motori elettrici sono stati preferiti ai servomotori e ai motori passo passo perché consentono al plotter di tracciare curve omogenee e prive di scalature. Per ottenere questi risultati, però, occorre un software di controllo molto sofisticato che, invece di generare e trasmettere una serie di impulsi, permetta di variare con continuità il voltaggio applicato ai motori.

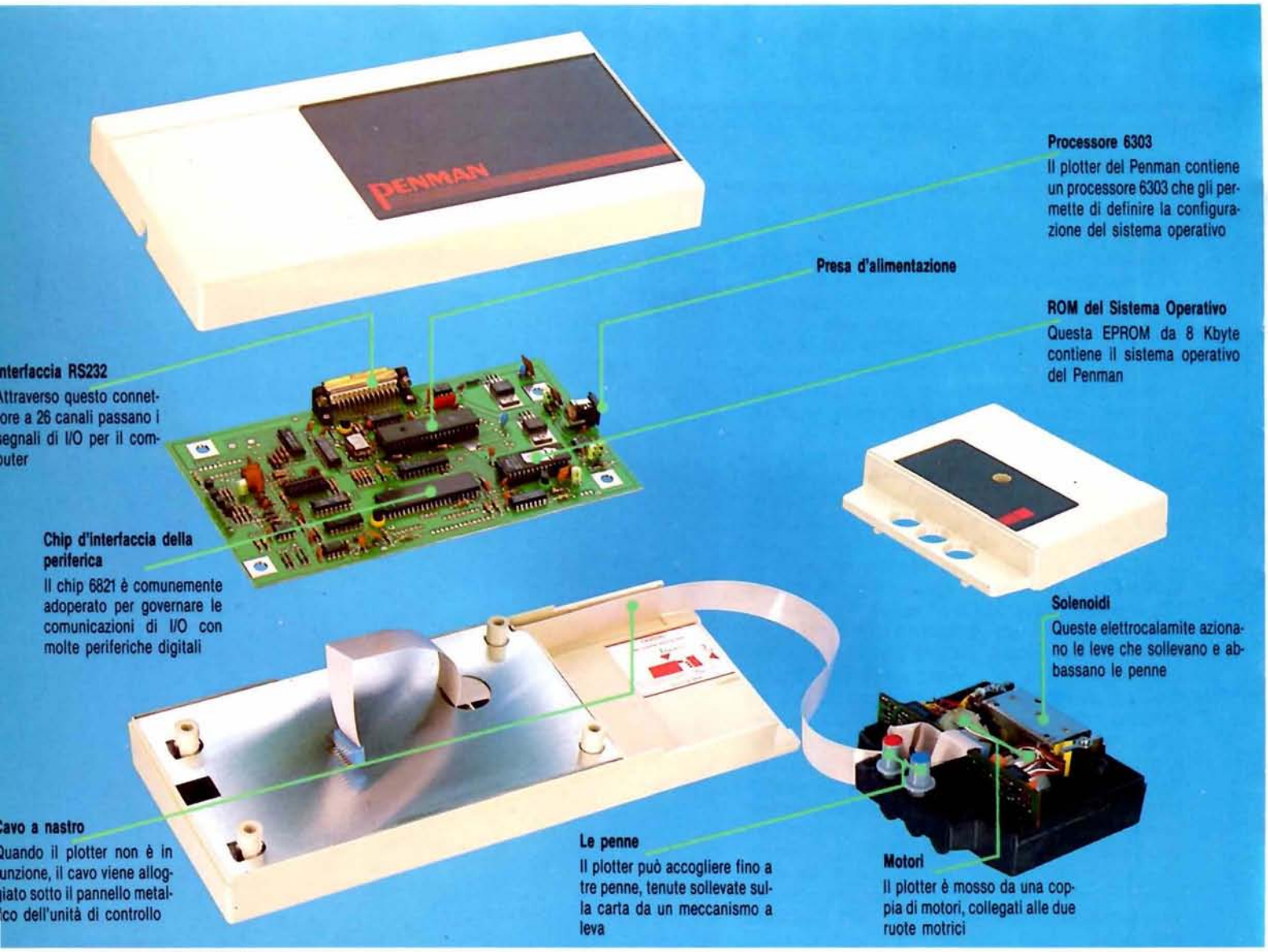


CHRIS STEVENS

La scheda del plotter distribuisce la tensione d'alimentazione, decodifica i segnali provenienti dall'unità di controllo e li trasmette alle elettrocalamite e ai motori. Sugli alberini dei motori è montato un disco stroboscopico, associato ad un rivelatore fotosensibile. Quando il disco stroboscopico gira, gli impulsi luminosi colpiscono il rivelatore. Questo comunica ai circuiti della scheda la velocità di rotazione del motore e permette al plotter di calcolare la propria posizione. Questo meccanismo si dimostra particolarmente utile quando il Penman viene usato come mouse, poiché il computer deve conoscere con

Una personalità poliedrica

Il Penman Plotter è composto da un'unità di controllo, collegata al computer con interfaccia RS232, e da una specie di tartaruga mobile. Il cavo che congiunge tartaruga e modulo è bidirezionale e permette alle due unità di comunicare tra loro. Quindi, il modulo può ordinare alla tartaruga di portarsi in una posizione e questa può comunicarla al sistema di controllo. Così è possibile usare il Penman Plotter non solo come tartaruga e plotter, ma anche come mouse



CHRIS STEVENS

esattezza la direzione e la velocità delle ruote motrici per spostare di conseguenza il cursore sullo schermo.

La scheda posta all'interno dell'unità di controllo contiene tre chip principali. Il processore 6303 (derivato dal 6800), che permette di lavorare con diversi sistemi operativi, incorpora una piccola RAM di lavoro nella quale viene registrata la posizione del plotter. Una ROM 6821 viene usata per interfacciare la periferica, infine, una EPROM contiene i programmi dimostrativi che il Penman esegue quando il cavo RS232C non è collegato.

Il Penman Plotter è un sistema polifunzionale. Il modo "simulatore di terminale", che permette di controllare l'apparecchio da tastiera, viene abilitato caricando direttamente il programma *Pentlk* oppure, se il plotter/robot viene usato come un mouse, selezionando il programma dal menu principale. Caricato il software, i comandi vengono trasmessi all'unità di controllo attraverso la porta RS232, sotto forma di codici ASCII. Il comando PRINT 'I' inizializza il Penman che, ricevuto il segnale, seleziona una delle tre possibili velocità di trasmissione (300, 1200, 9600 baud). Fissata la velocità di trasmissione, il comando H fa eseguire al plotter/robot una sequenza di movimenti standard che lo portano nella posizione di riposo, nell'angolo inferiore sinistro del foglio di carta.

Partendo da questa posizione, il Penman individua il margine inferiore del foglio usando i sensori di luce, quindi ruota di 90° ed esegue la stessa operazione sul margine sinistro. Infine assume la posizione di partenza definitiva posizionandosi con precisione sull'angolo inferiore sinistro del foglio (la penna centrale dista esattamente 50 mm. da entrambi i margini). Per accentuare il contrasto di luminosità tra la carta e il piano di lavoro, la Penman Products ha corredato l'apparecchio di un cartoncino nero sul quale appoggiare il foglio da disegno.

Il programma applicativo *Pentlk* può essere eseguito in modo diretto (un comando alla volta) oppure concatenando una serie di comandi in modo da formare un programma

che può essere caricato in memoria, salvato o elaborato.
Nel modo "simulatore di terminale" il movimento del robot sul foglio è di tipo cartesiano. Quando il plotter riceve l'ordine di spostarsi alla posizione (500,500), si muove direttamente verso il punto indicato dalle coordinate invece di eseguire 500 passi in entrambe le direzioni. Un foglio di carta A4 possiede al massimo 2100 coordinate sull'asse x e 2970 sull'asse y. I comandi possono essere introdotti in modo "assoluto" o "relativo": nel primo caso vengono preceduti dalla lettera A, nel secondo dalla lettera R. Le penne vengono azionate dai seguenti comandi di tipo LOGO: P seleziona una penna, U la solleva e D la abbassa.
Il Penman Plotter può eseguire nel modo diretto anche i movimenti della grafica della tartaruga, ma i comandi appaiono più complicati di quelli cartesiani perché la misura della distanza da percorrere deve essere espressa in notazione esadecimale, preceduta dal carattere $. Questo tipo di grafica richiede la notazione esadecimale perché consulta direttamente i bit degli indirizzi di controllo all'interno del computer, saltando a piè pari il software che normalmente decodifica le coordinate cartesiane. Comunque il Penman Plotter offre un secondo sistema per tracciare la grafica della tartaruga: il suo software applicativo, infatti, contiene alcuni programmi per controllare il robot direttamente in LOGO.

Controllo nel modo "robot"

Un sistema di controllo diretto simile al precedente è quello del modo "robot", che consente di pilotare il plotter del Penman attraverso il registro dei dati di direzione (DDR) della porta utente. Ogni bit del registro controlla un differente aspetto dei movimenti del robot: i bit 0 e 1 controllano il motore di destra, i bit 2 e 3 quello di sinistra. I bit 4 e 5 accendono e spengono i motori, mentre il bit 6 dirige il movimento verso l'alto e il basso della penna. Il Penman, a sua volta, invia al registro dei dati una serie di informazioni relative al proprio funzionamento, ad esempio: gli errori di posizionamento oppure l'avvenuta individuazione del margine della pagina tramite i sensori di luce.
Il Penman Plotter accetta anche una discreta gamma di comandi per la riproduzione di scritte. Tali comandi e il loro funzionamento sono assai simili a quelli impiegati sulle stampanti/plotter dei micro più comuni (vedere a pag. 1211). L'altezza dei singoli caratteri può variare da 1 a 127 mm. e i testi possono essere stampati nelle quattro diverse direzioni. Infine, contrariamente alle stampanti plotter di tipo convenzionale, il Penman Plotter consente anche di scrivere testi in obliquo.
Il manuale d'uso del plotter contiene una lista completa dei comandi disponibili, accompagnati da brevi spiegazioni sul loro funzionamento, e una descrizione dell'hardware. Quest'ultima è forse troppo tecnica e fa pensare che il Penman Plotter sia indicato principalmente per gli utenti che già possiedono una discreta conoscenza del funzionamento del loro computer.
Il Penman Plotter può essere considerato un passo avanti nello sviluppo di una tartaruga/plotter polivalente; se viene usato in modo corretto, la sua capacità risolutiva è paragonabile a quella dei plotter comunemente usati negli studi di grafica pubblicitaria. Nonostante ciò, i programmi di corredo presentano caratteristiche spiccatamente didattiche, che favoriscono la comprensione e l'applicazione dei principi base del software "robotico".
Sebbene il mercato offra apparecchi a prezzi più contenuti, il Penman Plotter, grazie alla sua ampia flessibilità di prestazioni, rappresenta un'interessante proposta per le scuole, dove può essere usato come sussidio didattico per le più svariate applicazioni.

PENMAN PLOTTER

DIMENSIONI

mm. 335 x 128 x 55

INTERFACCE

Interfaccia RS232C, che permette il collegamento con qualsiasi computer

RISOLUZIONE

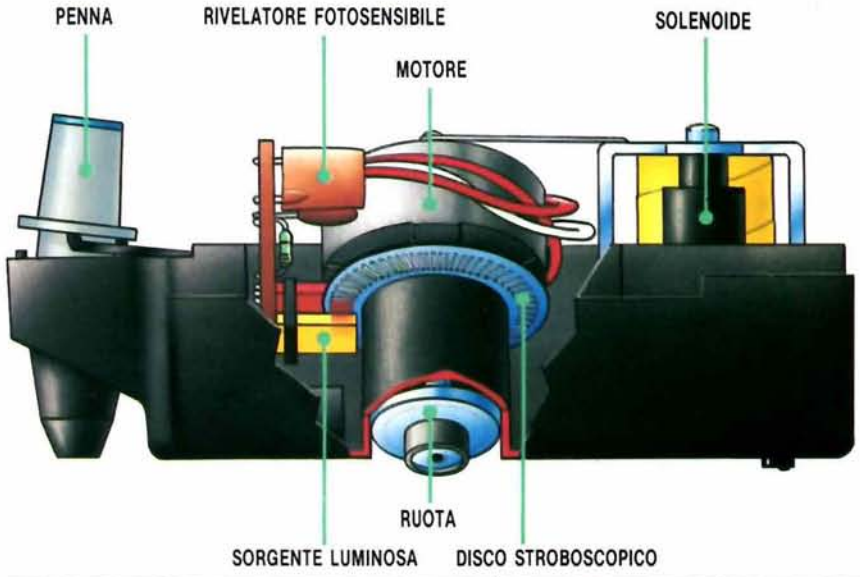
Dimensione per i testi:
1 a 127 mm.
Risoluzione grafica: 0,1 mm.

DOCUMENTAZIONE

Il manuale fornisce molti dettagli tecnici che mettono in grado i più esperti di sfruttare appieno le potenzialità del Penman. Manca una parte specificamente indirizzata ai principianti

IL SISTEMA DI POSIZIONAMENTO

L'unità di controllo del Penman riceve i dati relativi alla posizione del plotter da un sistema a feedback, interno al robot, che ne misura gli spostamenti o "passi" per mezzo di un disco stroboscopico fissato agli alberini di trasmissione delle ruote motrici.
Al disco è abbinato uno strumento composto da una sorgente luminosa e da un rivelatore fotosensibile. Quando le ruote girano, il disco produce un effetto stroboscopico trasformando la luce che lo colpisce in una serie di segnali luminosi che, a loro volta, colpiscono il rivelatore. Il numero dei segnali ricevuti dal rivelatore permette di calcolare i passi eseguiti dal plotter



KEVIN JONES

Navigatori solitari

Aggiungiamo altri imprevisti al viaggio verso il Nuovo Mondo

Gli eventi secondari che si verificano durante la traversata vengono trattati da brevi sezioni di programma, selezionate in modo casuale dalla subroutine alla linea 5500. Ai primi 5 eventi, già descritti, ne vanno aggiunti altri 8 per un totale di 13. È quindi necessario modificare RM alla linea 46, inserendovi il valore 13, e aggiungere i numeri di linea delle nuove routine alla linea 5525. Per ogni settimana di viaggio il programma principale seleziona a caso da questa lista un evento secondario:

```
5525 ON X GOTO 5540,5570,5570,5570,5570,
5600,5700,5800,5850,5900,5950,6000,6050
```

Il primo numero di linea da aggiungere alla linea 5525 è 5600, da cui inizia la routine per la cattura del pesce. Se il generatore di eventi casuali assegna a X il valore 6, il controllo del programma passa a questa routine. Quando un membro dell'equipaggio muore, il decesso viene registrato nella variabile CN solo alla fine della settimana. Tuttavia può accadere che durante la settimana muoia l'intero equipaggio, nel qual caso la routine della pesca non deve essere eseguita e il programma verifica tale eventualità eseguendo il ciclo alla linea 5610. Questo esamina la matrice dell'equipaggio e controlla se il livello di rendimento di qualche uomo ha assunto il valore −999 durante la settimana, ossia se c'è stato qualche decesso. La linea 5630 calcola il numero dei marinai morti e lo sottrae dal totale dell'equipaggio. Se il risultato è minore di 1, la routine viene ignorata e il controllo torna al programma principale.
In caso contrario, la linea 5650 genera un numero casuale compreso tra 11 e 20 che rappresenta, in chili, la quantità di pesce catturato. Tale valore viene sommato alla carne ancora disponibile (linea 5680) e il programma comunica al giocatore per quanto tempo dureranno le scorte di carne. Infine la linea 5685 divide il totale per il numero dei membri dell'equipaggio e le loro necessità settimanali ed esegue una nuova stima del numero di settimane che le scorte dovrebbero coprire.
L'evento successivo è una tempesta. A partire dalla linea 5700, l'equipaggio cerca di raccogliere la pioggia con dei barili, per rinnovare le provviste d'acqua. Il programma genera un numero casuale compreso tra 11 e 20 (linea 5735), che indica il numero di barili d'acqua raccolti, e lo somma al numero dei barili rimasti (linea 5750). Infine la linea 5755 calcola e visualizza la nuova stima della durata delle scorte d'acqua.

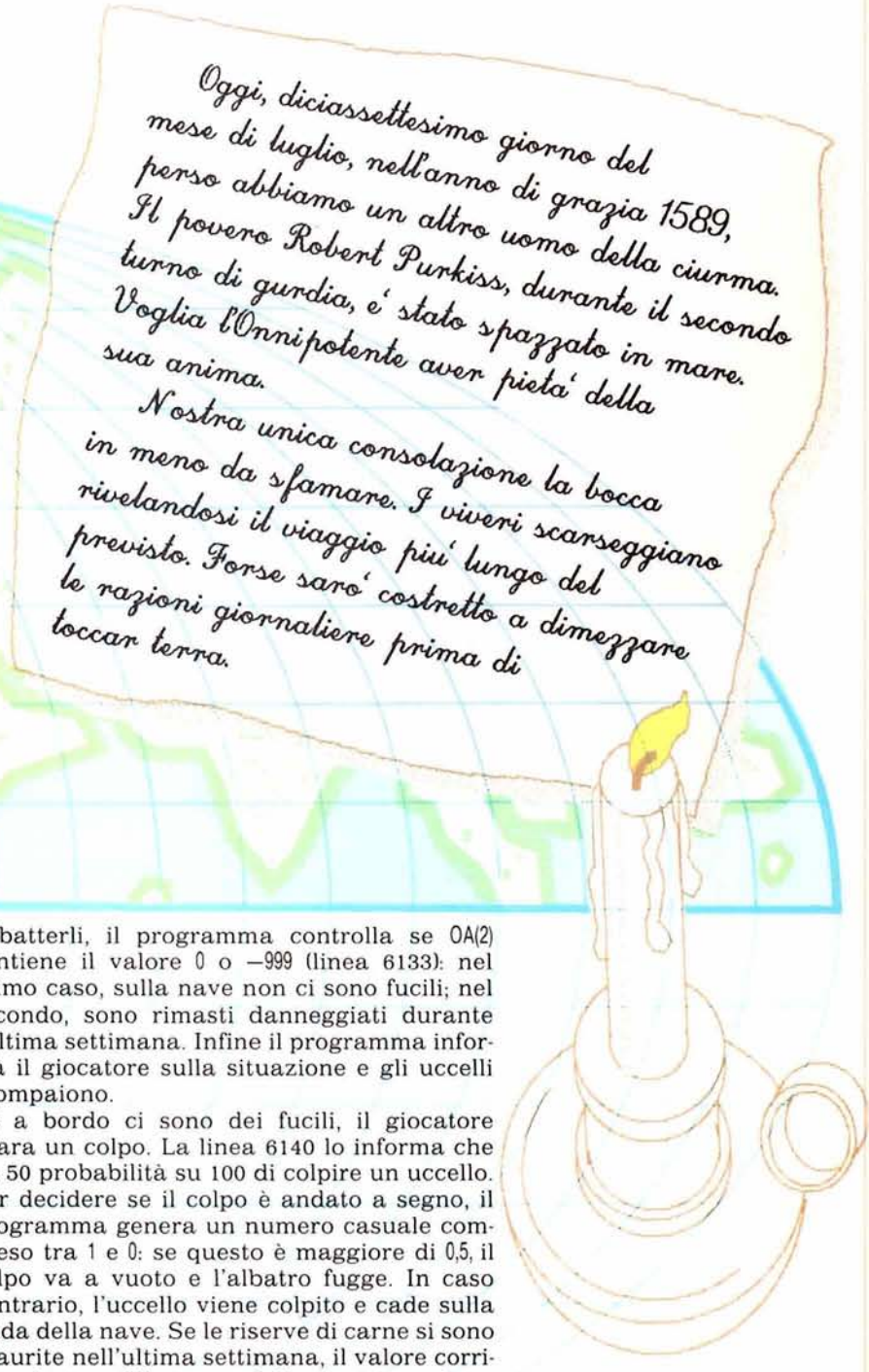
L'ottavo evento riguarda i venti favorevoli che aumentano la velocità della nave e riducono la durata del viaggio di mezza settimana; questa viene sottratta dal totale delle settimane alla linea 5835. La routine tempo bello inizia alla linea 5850 e corrisponde al nono indirizzo nella linea 5525. Questo evento permette all'equipaggio di riposarsi e di rimettersi in salute, facendo aumentare il livello di rendimento nella matrice TS(,) controllato dal ciclo alla linea 5882. Per evitare che cresca anche il rendimento degli uomini deceduti (che, paradossalmente tornerebbero in vita!), la linea 5884 verifica se il valore corrente del rendimento di ogni membro dell'equipaggio è 0 o −999: in questo caso passa al membro successivo. Quindi la linea 5886 aggiunge al rendimento di ogni uomo ancora in vita un numero casuale compreso tra 5 e 15.
Quando la linea 5525 seleziona il numero d'evento 10, il programma salta alla linea 5900, la subroutine che causa una perdita di medicinali. Il cattivo tempo danneggia la metà delle bottiglie di medicinali e ciò impedisce di curare adeguatamente i membri dell'equipaggio malati. Per prima cosa il programma verifica se ci sono ancora medicine a bordo, esaminando il primo elemento della matrice delle scorte OA(). Se il suo valore è 0 oppure −999, le bottiglie di medicinali sono terminate e l'evento 10 viene ignorato. Se a bordo ci sono ancora medicinali, la linea 5925 ne dimezza la quantità e calcola il risultato (naturalmente un numero intero). Quindi il programma visualizza un messaggio che comunica al giocatore quante bottiglie sono rimaste.
Nell'undicesimo evento una forte ondata danneggia alcune armi da fuoco, facendole arrugginire. Le armi sono importanti perché servono per difendersi, per procacciarsi il cibo o anche come merce di scambio. La routine per le armi arrugginite è simile a

Auftralem

Terra del Fuego

RRA AVSTRALIS

quella per le bottiglie di medicinali. Verifica se ci sono ancora armi a bordo, ne dimezza la quantità (linea 5955) e visualizza il numero di armi rimaste come intero (linea 5975).
Nel dodicesimo evento le pezze di stoffa vengono mangiate dai topi e la routine alla linea 6000 riduce il loro numero della metà. Poiché la stoffa è il quarto elemento della matrice delle merci OA( ), il programma esamina OA(4) per controllare se ci sono pezze nella stiva (linea 6005). Se queste erano state acquistate all'inizio del viaggio, i topi "sbucano fuori" dalla linea 6025 e ne distruggono la metà. Quindi il programma comunica al giocatore il numero di pezze rimaste.
Si noti che a questo punto è possibile facilitare o complicare le successive fasi di gioco modificando i parametri dei diversi eventi. Infatti il giocatore può ampliare o restringere i margini di selezione dei numeri casuali e accrescere o ridurre le quantità di merci che vanno perdute.



Avvistamento degli albatro

La tredicesima e ultima routine è l'avvistamento degli albatro. Questo evento si differenzia dai precedenti perché può verificarsi due volte in una stessa settimana. Se l'istruzione ON X GOTO seleziona il numero 13, l'avvistamento degli albatro segue il suo corso normale come un qualsiasi evento secondario. Se le scorte di carne scarseggiano (razioni dimezzate) quando vengono avvistati gli albatro, l'equipaggio può cercare di abbattere gli uccelli e usarli come cibo. Nel programma principale, gli eventi secondari vengono richiamati dall'istruzione GOSUB alla linea 860, mentre quelli principali iniziano alla linea 870. La seconda possibilità di avvistare gli albatro si verifica alla linea 875. Questa controlla se l'equipaggio è a mezza razione di carne, verificando se HR(3) è uguale a 0,5. In questo caso un valore casuale, generato dalle istruzioni AND RND(1)<,5, offre una probabilità di avvistamento del 50%. Se tale probabilità si realizza, il controllo del programma passa alla subroutine albatro che inizia alla linea 6050.
Quando compaiono gli albatro, alla variabile A$ viene assegnato il valore "Y" per indicarne l'avvistamento. Questo è un segno di buon auspicio, che in seguito permetterà di ridurre i rischi di ammutinamento. La linea 6075 controlla se la carne scarseggia, verificando se la quantità disponibile è uguale o maggiore rispetto alle necessità settimanali degli uomini moltiplicate per la rimanente durata del viaggio. Se le scorte sono sufficienti, gli albatro volano via e la linea 6130 restituisce il controllo al programma principale.
Se invece la carne rimasta è insufficiente, il programma informa il giocatore che gli uccelli pesano 10 chili e gli chiede se vuole provare ad abbatterli. La linea 6115 controlla se la risposta è positiva leggendone il primo carattere: se il giocatore decide di non sparare, gli uccelli volano via. Se al contrario decide di abbatterli, il programma controlla se OA(2) contiene il valore 0 o −999 (linea 6133): nel primo caso, sulla nave non ci sono fucili; nel secondo, sono rimasti danneggiati durante l'ultima settimana. Infine il programma informa il giocatore sulla situazione e gli uccelli scompaiono.
Se a bordo ci sono dei fucili, il giocatore spara un colpo. La linea 6140 lo informa che ha 50 probabilità su 100 di colpire un uccello. Per decidere se il colpo è andato a segno, il programma genera un numero casuale compreso tra 1 e 0: se questo è maggiore di 0,5, il colpo va a vuoto e l'albatro fugge. In caso contrario, l'uccello viene colpito e cade sulla tolda della nave. Se le riserve di carne si sono esaurite nell'ultima settimana, il valore corrispondente, all'interno della matrice, viene portato da −999 a 0 (linea 6160) per consentire la registrazione del peso dell'albatro abbattuto. Altrimenti la nuova scorta di carne (10 kg.) viene sommata all'elemento PA(3) della matrice delle provviste e viene visualizzato il nuovo totale.
Purtroppo, secondo le credenze dei marinai, la morte dell'albatro può essere di cattivo auspicio. Questo sfavorevole presagio, segnalato dal programma assegnando "Y" alla variabile B$, si contrappone a quello predisposto dalla linea 6055. Gli effetti concreti di questa sorte avversa si manifestano in seguito: senza che il giocatore ne venga informato, vengono improvvisamente modificati i parametri degli eventi secondari per ostacolare il cammino della nave!

IAN McKINNELL SUL MAKINTOSH

MODULO SETTE: ALTRI EVENTI CASUALI

Da aggiungere al ciclo principale del viaggio

```
875 IF HR(3) = .5 AND RND(1) < .5 THEN PRINT
    CHR$(147): GOSUB 6050
```

Subroutine per gli altri eventi casuali

```
5600 REM EVENTO 6 - CATTURA DEL PESCE
5605 X=0
5610 FOR T = 1 TO 16
5615 IF TS(T,2) = -999 THEN X = X+1
5620 REM CONTEGGIO DECESSI DELLA SETTIMANA
     CORRENTE
5625 NEXT
5630 IF CN - X < 1 THEN RETURN
5635 REM NON PROSEGUE SE TUTTO L'EQUIPAGGIO E'
     MORTO
5640 PRINT
5645 S$="DURANTE LA SETTIMANA*": GOSUB 9100
5646 PRINT: GOSUB 9200
5650 S$="UNO DEGLI UOMINI HA PESCATO*":
     GOSUB 9100
5655 X = INT (RND(1)*10) + 11
5660 REM DA 10 A 20 CHILI
5662 PRINT X; "CHILI DI PESCE"
5665 PRINT: GOSUB 9200
5670 S$="SCORTE DI CARNE*": GOSUB 9100
5675 S$="SUFFICIENTI PER CIRCA*": GOSUB 9100
5678 IF PA(3) = -999 THEN PA(3) = 0
5680 PA(3) = PA(3) + X
5685 PRINT INT(PA(3)/(CN*PN(3)));" SETTIMANE"
5690 GOTO 5530
5700 REM EVENTO 7 - TEMPESTA DI PIOGGIA
5705 PRINT
5710 S$="DURANTE LA SETTIMANA*": GOSUB 9100
5715 PRINT: GOSUB 9200
5720 S$="UNA TEMPESTA PERMETTE*": GOSUB 9100
5725 S$="UNA RACCOLTA D'ACQUA *": GOSUB 9100
5730 PRINT: GOSUB 9200
5735 X = INT (RND(1)*10) + 11
5736 REM DA 10 A 20 BARILI
5740 S$="NUOVE SCORTE D'ACQUA *": GOSUB 9100
5745 S$="SUFFICIENTI PER CIRCA *": GOSUB 9100
5748 IF PA(4) = -999 THEN PA(4) = 0
5750 PA(4) = PA(4) + X
5755 PRINT INT (PA(4)/(CN*PN(4)));" SETTIMANE"
5760 GOTO 5530
5800 REM EVENTO 8 - VENTI FAVOREVOLI
5805 PRINT
5808 S$="VENTI FAVOREVOLI*": GOSUB 9100
5810 S$="PER TUTTA LA SETTIMANA*": GOSUB 9100
5815 PRINT: GOSUB 9200
5820 S$="LA VELOCITA' AUMENTA E LA*":
     GOSUB 9100
5825 S$="DURATA DEL VIAGGIO CALA *": GOSUB 9100
5830 S$="DI MEZZA SETTIMANA!*": GOSUB 9100
5835 EW = EW - .5
5839 GOTO 5530
5850 REM EVENTO 9 - TEMPO BELLO
5855 PRINT
5860 S$="BEL TEMPO PER TUTTA*": GOSUB 9100
5862 S$="LA SETTIMANA!!*": GOSUB 9100
5865 PRINT: GOSUB 9200
5870 S$="IL MORALE MIGLIORA E GLI*": GOSUB 9100
5875 GOSUB 9200
5880 S$="UOMINI RECUPERANO FORZE*": GOSUB 9100
5882 FOR T = 1 TO 16
5884 IF TS(T,2) = 0 OR TS(T,2) = -999 THEN 5888
5886 TS(T,2) = TS(T,2) + INT (RND(1)*11) + 5
5888 NEXT
5889 GOTO 5530
5900 REM EVENTO 10 - PERDITA DEI MEDICINALI
5905 IF OA(1) = 0 OR OA(1) = -999 THEN RETURN
5910 PRINT
5915 S$="META' DELLE BOTTIGLIE DI*": GOSUB 9100
5920 S$="MEDICINALE SI SONO ROTTE*": GOSUB 9100
5925 OA(1) = INT(OA(1)/2)
5930 PRINT: GOSUB 9200
5935 S$="RESTANO SOLTANTO*": GOSUB 9100
5940 PRINT OA(1);" BOTTIGLIE"
5945 GOTO 5530
5950 REM EVENTO 11 - FUCILI ARRUGGINITI
5955 IF OA(2) = 0 OR OA(2) = -999 THEN RETURN
5960 PRINT
5965 S$="META' DEI FUCILI SI*": GOSUB 9100
5970 S$="SONO ARRUGGINITI E*": GOSUB 9100
5972 S$="NON FUNZIONANO PIU'!!*": GOSUB 9100
5975 OA(2) = INT(OA(2)/2)
5980 PRINT: GOSUB 9200
5985 S$="RESTANO SOLTANTO*": GOSUB 9100
5990 PRINT OA(2);" FUCILI FUNZIONANTI"
5995 GOTO 5330
6000 REM EVENTO 12 - PERDITA DELLE BALLE DI
     STOFFA
6005 IF OA(4) = 0 OR OA(4) = -999 THEN RETURN
6010 PRINT
6015 S$="I TOPI HANNO ROSICCHIATO*": GOSUB 9100
6020 S$="META' DELLE BALLE DI*": GOSUB 9100
6022 S$="STOFFA DANNEGGIANDOLE*": GOSUB 9100
6024 S$="IRREPARABILMENTE*": GOSUB 9100
6025 OA(4) = INT (OA(4)/2)
6030 PRINT: GOSUB 9200
6035 S$="RESTANO SOLTANTO*": GOSUB 9100
6040 PRINT OA(4); " BALLE"
6045 GOTO 5530
6050 REM EVENTO 13 - ALBATRO
6055 PRINT: A$ = "S"
6060 S$="UN ALBATRO VOLA SULLA NAVE*": GOSUB 9100
6062 GOSUB 9200
6065 S$="E' UN PRESAGIO FAVOREVOLE*": GOSUB 9100
6068 S$="CHE SOLLEVA IL MORALE*": GOSUB 9100
6069 S$="DELL'EQUIPAGGIO*": GOSUB 9100
6070 PRINT: GOSUB 9200
6075 IF PA(3) < (CN*PN(3)*(JL-WK+1)) THEN 6090
6080 REM LA CARNE NON MANCA
6085 GOTO 6122
6090 S$="LE SCORTE DI CARNE SONO*": GOSUB 9100
6092 S$="QUASI ESAURITE E*": GOSUB 9100
6095 S$="L'ALBATRO PESA 10 KG!*": GOSUB 9100
6100 PRINT: GOSUB 9200
6105 S$="VUOI ABBATTERLO (S/N)?*": GOSUB 9100
6110 INPUT I$
6112 PRINT: GOSUB 9200
6115 IF LEFT$(I$,1)="S" THEN 6133
6120 S$="FORSE E' UNA COSA SAGGIA!*":
     GOSUB 9100
6122 PRINT: GOSUB 9200
6125 S$="L'ALBATRO VOLA VIA...*": GOSUB 9100
6130 GOTO 5330
6133 IF OA(2) = 0 OR OA(2) = -999 THEN 6180
6135 S$="SPARI UN COLPO..... *": GOSUB 9100
6138 GOSUB 9200: GOSUB 9200
6140 IF RND(1) < .5 THEN 6150
6145 S$=" ...MA SENZA SUCCESSO!*: GOSUB 9100
6148 GOTO 6122
6150 S$="L'ALBATRO CADE SULLA TOLDA*":
     GOSUB 9100
6155 PRINT: GOSUB 9200
6160 IF PA(3) = -999 THEN PA(3) = 0
6162 PA(3) = PA(3) + 10: B$ = "S"
6165 S$="ADESSO POSSIEDI 10 CHILI*": GOSUB 9100
6167 S$="DI CARNE... IN PIU'*": GOSUB 9100
6170 S$="MA DA ORA LA FORTUNA*": GOSUB 9100
6172 S$="PUO' VOLTARTI LE SPALLE!*": GOSUB 9100
6174 GOTO 5530
6180 S$="NON HAI NEMMENO UN FUCILE!*":
     GOSUB 9100
6190 GOTO 6122
```

Basic & Basic

Spectrum

Apportare le seguenti modifiche:

```
875 IF HR(3)=.5ANDRND(1)<.5THENCLS:GO SUB 6050
5527 IF X=6 THEN GO TO 5600
5528 IF X=7 THEN GO TO 5700
5529 IF X=8 THEN GO TO 5800
5530 IF X=9 THEN GO TO 5850
5531 IF X=10 THEN GO TO 5900
5532 IF X=11 THEN GO TO 5950
5533 IF X=12 THEN GO TO 6000
5534 IF X=13 THEN GO TO 6050
5535 PRINT:S$=K$:GO SUB 9100
5536 LET I$=INKEY$:IF INKEY$="" THEN GO TO 5536
6115 IF I$(1 TO 1)="S" THEN GO TO 6133
```

Micro BBC

Apportare le seguenti modifiche:

```
875 IF HR(3)=.5AND RND(1)<.5 THEN CLS:GOSUB6050
```

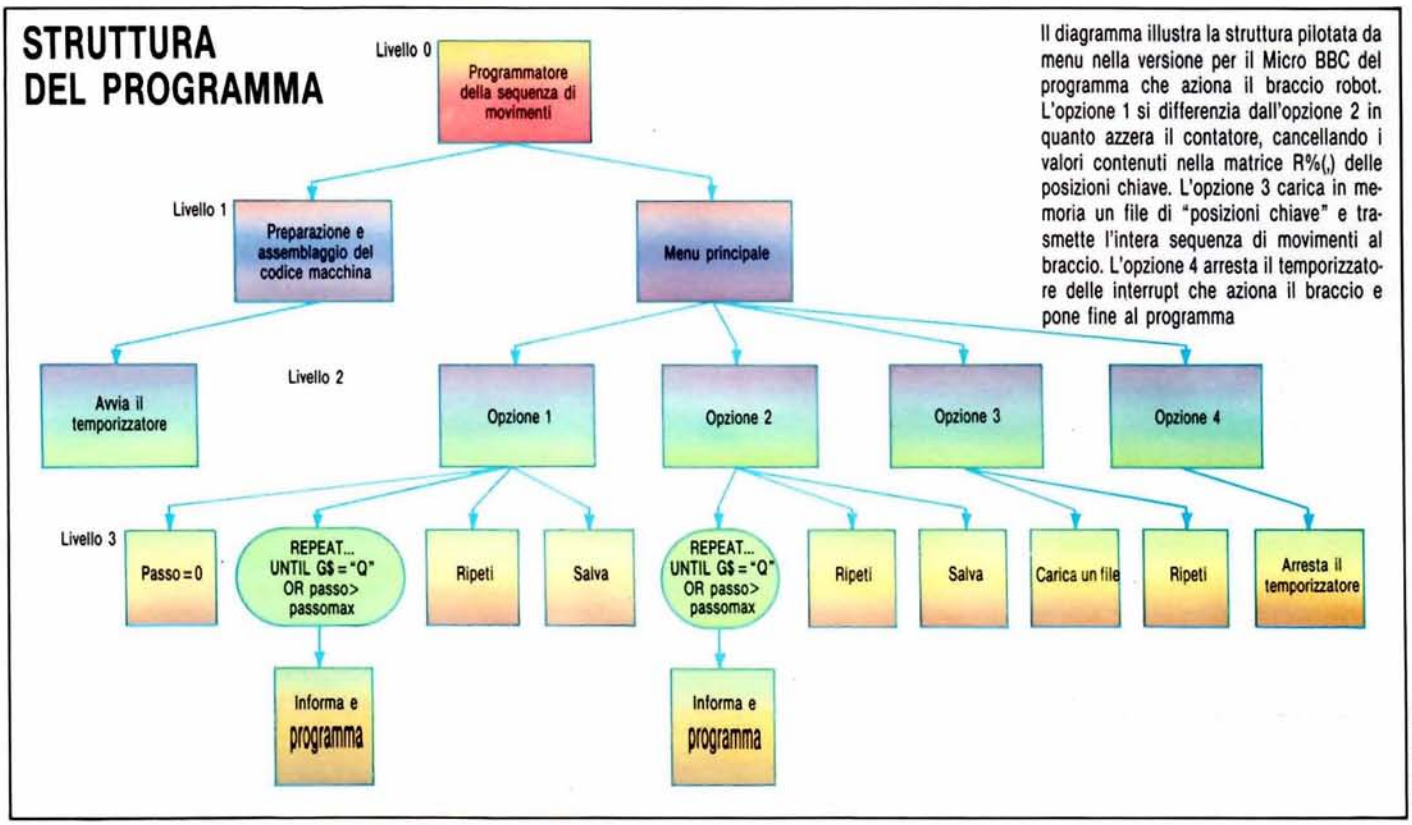
A portata di mano

Realizziamo il software per il controllo del braccio robot

I quattro motori che muovono il braccio robot sono collegati ad un computer e da esso controllati attraverso quattro linee per i dati. Nel caso del Micro BBC, le linee originano dai 4 piedini, da D0 a D3, della porta utente. Poiché i servomotori vengono azionati da una sequenza continua di impulsi che ne determinano l'angolo di rotazione, non possono essere controllati esclusivamente dal BASIC, ma implicano l'impiego di un programma "cuneo" in codice macchina. Questo trasmette ai motori una nuova serie d'impulsi ogni sessantesimo di secondo, ossia ogni qual volta viene generata un'interrupt (per i principi di base di questa tecnica di controllo vedere alle pagine 1392 e 1403). Il programma per il controllo multiplo dei servomotori, a pag. 1405, prevede l'uso diretto della tastiera per verificare il funzionamento complessivo del braccio e dei singoli motori. Poiché la routine in codice macchina viene eseguita "in sottofondo", è possibile interrompere il programma BASIC per la scansione della tastiera e inserire i numeri direttamente nelle locazioni che controllano l'angolazione dei motori. L'indirizzo della prima locazione di memoria corrisponde ad angolo: il valore in essa contenuto indica l'angolo di rotazione del motore 0. L'indirizzo angolo+1 seleziona la posizione del motore 1 e così via. Il motore 0 controlla i movimenti della base del braccio, il motore 1 la rotazione della spalla e il motore 3 quella del gomito. Il motore 4, infine, apre e chiude il meccanismo di presa. Se l'angolo di rotazione viene incrementato o decrementato di una sola unità, i motori rispondono subito al comando. Se invece le rotazioni sono più ampie (valori compresi tra 0 e 128), i motori entrano in azione dopo un intervallo di tempo. Il braccio robot conclude ogni spostamento con una leggera vibrazione, dovuta all'effetto combinato dell'inerzia e dell'attrito delle articolazioni.

Programmazione della sequenza di movimento

Per collaudare il sistema è sufficiente un semplice programma di controllo, ma per l'impiego pratico del braccio robot occorre un programma molto più sofisticato. Indubbiamente la programmazione del software di controllo è la fase più complessa dell'intero progetto. Molti robot industriali vengono programmati azionando manualmente la presa secondo la sequenza di movimenti che dovrà ripetere. Durante questa operazione, il braccio registra automaticamente le posizioni chiave della sequenza. Sfortunatamente questa tecnica richiede apparecchiature piutto-



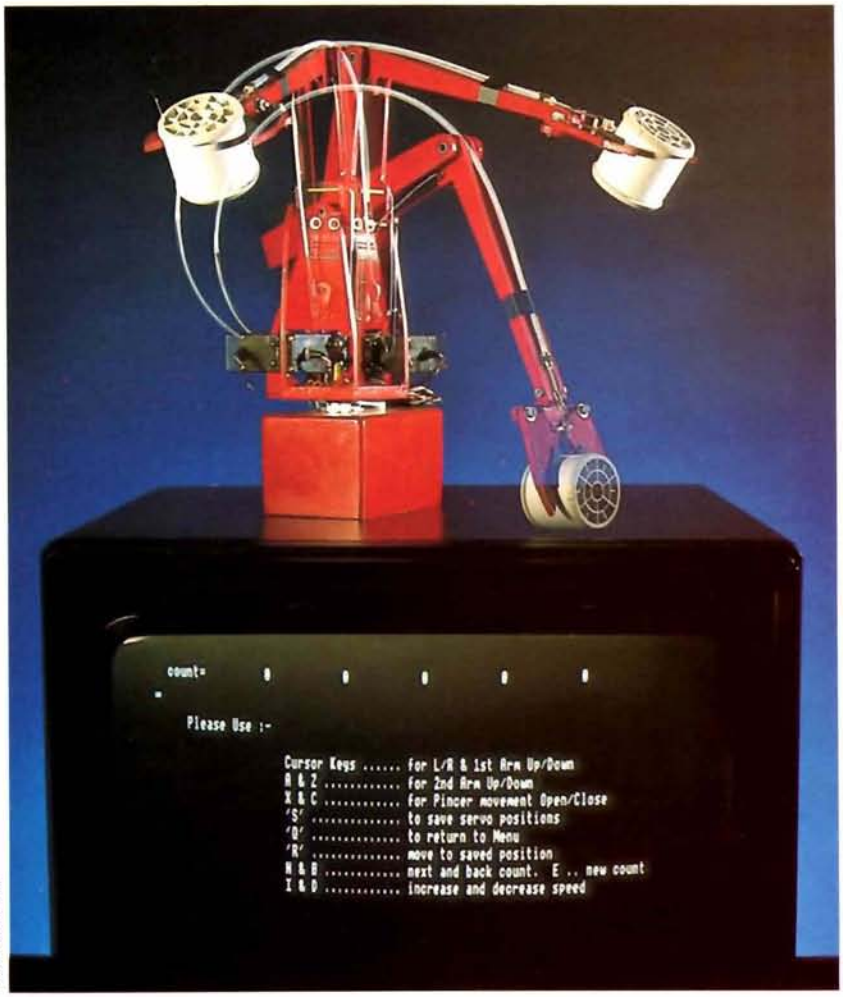
Il diagramma illustra la struttura pilotata da menu nella versione per il Micro BBC del programma che aziona il braccio robot. L'opzione 1 si differenzia dall'opzione 2 in quanto azzera il contatore, cancellando i valori contenuti nella matrice R%(,) delle posizioni chiave. L'opzione 3 carica in memoria un file di "posizioni chiave" e trasmette l'intera sequenza di movimenti al braccio. L'opzione 4 arresta il temporizzatore delle interrupt che aziona il braccio e pone fine al programma

KEVIN JONES

sto sofisticate, quali sensori angolari a feedback, che il nostro robot non possiede.
Ma anche in alcune applicazioni industriali, laddove i robot operano in ambienti pericolosi che non permettono all'uomo di intervenire direttamente per programmare il braccio, viene usata una tecnica di programmazione alternativa che consiste nel controllare il braccio a distanza, tramite la tastiera. Si ordina così al robot di eseguire una serie di movimenti, di salvare le posizioni chiave della sequenza, premendo un apposito tasto, e di registrare in una matrice le angolazioni dei quattro motori relative a ogni posizione. Questo processo consente di ripetere l'intera sequenza ritrasmettendo le posizioni memorizzate nella matrice con un adeguato fattore d'intervallo. Le sequenze possono essere salvate o caricate dalla memoria esterna (disco o nastro) usando un file sequenziale che contiene i dati registrati nella matrice. Questo è il metodo usato per il braccio robot.
La prima fase della progettazione del software è dedicata alla definizione dei tasti per il controllo a distanza dei movimenti del braccio. Sul Micro BBC, è possibile riprogrammare i tasti del cursore e adoperarli per controllare il movimento destra/sinistra della base del braccio e il movimento su/giù del braccio inferiore. I tasti A e Z muovono verso l'alto e il basso il braccio superiore mentre i tasti X e C aprono e chiudono le tenaglie della presa. I tasti I e D permettono di aumentare o ridurre la velocità del braccio, consentendo di eseguire più lentamente le manovre di precisione e più rapidamente i movimenti di maggior ampiezza. Il tasto S registra nella matrice la posizione corrente del braccio, mentre R riporta il braccio nella posizione memorizzata. Una sequenza può essere modificata rieseguendola, in uno dei due sensi, un passo alla volta. I tasti N e B controllano rispettivamente l'esecuzione diretta e inversa, mentre E interviene sulle singole posizioni chiave. Agendo su questi tre tasti è possibile cambiare ogni sequenza di movimenti preprogrammati. Le procedure informa e programma forniscono un elenco delle funzioni dei tasti, oltre ad effettuare una continua scansione della tastiera. Le procedure ripeti e salva consentono di ripetere o di salvare su dischetto o nastro la sequenza registrata nella matrice delle posizioni chiave. La procedura caricafile assegna alla matrice R%(,) i valori salvati in precedenza su un file sequenziale. Nel suo complesso, il programma è un completo strumento di programmazione, correzione e ripetizione delle sequenze di movimento del braccio robot.

Al lavoro...

Il braccio robot è stato realizzato partendo con uno specifico obiettivo: sollevare e spostare un oggetto delle dimensioni di un rocchetto di filo. Il codice che controlla il movimento del robot permette di programmare il braccio per questa operazione e di salvare la sequenza dei movimenti su un file. Quest'ultimo può essere ricaricato in memoria ed eseguito in qualsiasi momento



IAN McKINNELL

Un movimento più lineare

La routine in codice macchina che gestisce le interrupt controlla le angolazioni dei servomotori esaminando le rispettive locazioni a partire dall'indirizzo angolo. Ogni locazione contiene il valore relativo alla rotazione di ciascun motore. I bruschi movimenti generati dal programma di controllo (vedere a pag. 1405) sono un effetto delle istruzioni POKE del BASIC, usate per inserire i valori nelle locazioni corrispondenti. I cambiamenti di angolazione di maggior entità producono un brusco movimento del servomotore, che cerca di assumere la nuova posizione nel minor tempo possibile. Per ottenere movimenti del braccio più lineari, conviene introdurre piccoli valori in serie nelle quattro locazioni angolo fino ad ottenere la rotazione richiesta. A questo fine viene impiegato un secondo gruppo di locazioni, che iniziano da npos%, in cui un programma in BASIC deposita i cambiamenti di angolazione. Ad esso è abbinata una breve routine in codice macchina che cambia i valori contenuti nelle quattro locazioni angolo fino ad uguagliare quelli contenuti nelle corrispondenti locazioni npos%. Infine azionaservo chiama il programma in codice macchina che esegue questa operazione.
Inserendo un ciclo di ritardo nella routine in codice macchina, è possibile ridurre o aumentare la velocità di posizionamento dei servomotori: basta modificare il valore che determina il ciclo di ritardo. La locazione &81 contiene questo fattore di ritardo, che può essere cambiato dal programma BASIC, all'inizio della procedura ripeti: questa modifica consente di eseguire a differenti velocità le sequenze di movimenti programmate in precedenza. Il codice macchina di questa nuova routine inizia alla linea 2000 del listato.

PROGRAMMATORE DI SEQUENZE PER IL MICRO BBC

```
 10 REM ********************************
 12 REM **                            **
 13 REM ** PROGRAMMATORE DI SEQUENZE **
 14 REM **        MICRO BBC           **
 15 REM **                            **
 16 REM ********************************
 20 :
 25 MODE 3
 30 PROCprepara
 40 CLS:PRINT TAB(5,6)"Menu : "
 50 PRINT TAB(25,10)"1.........Nuova sequenza di movimenti"
 60 PRINT TAB(25,12)"2.........Aggiunta nuovi movimenti"
 70 PRINT TAB(25,14)"3.........Ripete una sequenza (file)"
 80 PRINT TAB(25,16)"4.........Esce dal programma"
 90 G$=GET$:IF G$="1" THEN passo=0
100 IF G$="1" OR G$="2" THEN PROCinforma: PROCprogramma: PROCripete:
    PROCsalva
110 IF G$="3" THEN PROCcaricafile: PROCripete
120 IF G$="4" THEN CLS: PROCarrestatemp: END
130 GOTO 40
140
150
160 DEF PROCinforma:CLS
170 PRINTTAB(5,5)"Tasti di controllo :- "
180 PRINTTAB(20,7) "Tasti del cursore ..  D/S base - S/G Braccio Inf."
190 PRINTTAB(20) "A & Z ..............  S/G Braccio Sup."
200 PRINTTAB(20)"X  &  C ..............  Apertura/Chiusura Presa"
210 PRINTTAB(20)" S  ..................  Salva le  posiz.  dei servom."
220 PRINTTAB(20)" Q  ..................  per tornare al menu"
230 PRINTTAB(20)" R  ..................  per tornare alla posiz. salvata"
240 PRINTTAB(20)"N & B ..............  passo succ.  e prec. E ... nuovo
    passo"
250 PRINTTAB(20)"I & D ..............  accresce & riduce la velocita'"
260 PRINTTAB(2,2)"passo="passo"      ";:FOR I%=0TO3:PRINTR%(passo,I%);
    "   ";: NEXT: PRINT
270 ENDPROC
280
290
300 DEF PROCprogramma:?&81=1
310 REM ridefinisce i tasti del cursore
320 *FX4,1
330 DX=2
340 REPEAT: REM Esamina la tastiera
350 IF IN(89)=EY(-38)THEN DX=DX+1 : REM "I"
360 IF IN(89)=EY(-51)THEN DX=DX-1 : IF DX<1 THEN DX=1 :REM "D"
380 IF IN(89)=EY(-26)THEN npos%?0=(npos%?0)+DX: REM "FRECCIA SIN."
    servom. 0 a destra
390 IF IN(89)=EY(-122)THEN npos%?0=(npos%?0)-DX: REM "FRECCIA DES."
    servom. 0 a sinistra
400 IF IN(89)=EY(-58)THEN npos%?1=(npos%?1)+DX: REM "FRECCIA SU"
    servom. 1 in alto
410 IF IN(89)=EY(-42)THEN npos%?1=(npos%?1)-DX: REM "FRECCIA GIU'"
    servom. 1 in basso
420 IF IN(89)=EY(-66)THEN npos%?2=(npos%?2)-DX: REM "A" servom. 2 in alto
430 IF IN(89)=EY(-98)THEN npos%?2=(npos%?2)+DX: REM "Z" servom. 2 in basso
440 IF IN(89)=EY(-67)THEN npos%?3=(npos%?3)+DX*3:  REM "X" servom.  3 apre
    la presa
450 IF IN(89)=EY(-85)THEN npos%?3=(npos%?3)-DX*3:  REM "C" servom.  3
    chiude la presa
460 IF IN(89)=EY(-52)THEN FOR I%=0TO3:npos%?I%=R%(passo,I%):NEXT
    :REM "R" ripete la posizione
470 IF IN(89)=EY(-86)THEN passo=passo+1:IF passo>passomax THEN passo=0
480 IF IN(89)=EY(-101)THEN passo=passo-1:IF passo<0 THEN passo=passomax
490 IF IN(89)=EY(-35)THEN PRINTTAB (10,20)"DEFINIRE IL VALORE DEL passo";:
    INPUT passo: REM "E" introduce il nuovo passo
500 IF IN(89)=EY(-82) THEN FOR I%=0 TO Numservo%:R%(passo,I%)=npos%?I%:NEXT
    :passo=passo+1 :REM "S" salva la posizione al passo corrente
510 IF passo>ultpassma< THEN ultpassma<=passo
520 IF vpasso<>passo THEN PRINTTAB(2,2)"passo="passo"    ";:FOR I%=0
    TO 3: PRINTR% (passo,I%);"    ";:NEXT:PRINT                    "
530 vpasso=passo
540 FOR I%=0 TO 3: IF npos%?I%>200 THEN npos%?I%=200
550 IF npos%?I%<10 THEN npos%?I%=10
560 NEXT
570 PROCazionaservom
580 G$=IN(89)EY$(1): UNTIL G$="0" OR passo>passomax
590 *FX4,0
600 ENDPROC
610
620
630 DEF PROCripete
640 REPEAT
650 PRINTTAB(11,23) "Vuoi ripetere la sequenza dei movimenti? <S/N>
    R per ripetere"
660 G$=GET$
670 UNTIL G$="S" OR G$="N" OR G$="R"
680 IF G$="N" THEN ENDPROC
690 IF G$<>"R" THEN INPUTTAB(11,22)"Intervallo di ritardo (da
    1 a 255), poi premi <return>":rit%:IF rit%>255 OR rit%<1 GOTO 690
700 ?&81=rit%
710 FOR Numpos=0 TO ultpassma<:PRINTTAB(19,2)"Num. di posizione
    sequenziale = ";Numpos;"  ";:FOR servom%=0 TO Numservo%:
    npos%?servom%=R%(Numpos,servom%):NEXT:PROCazionaservom: NEXT
720 GOTO640
730
740
750 PROCazionaservom
760 CALLazionaservom: REM versione in cod. macchina
820 ENDPROC
830
831
840 DEF PROCsalva: CLS
850 PRINT  TAB  (15,10)"Vuoi salvare la sequenza su  un  file ?<S/N>:
    IF GET$<>"S" THEN ENDPROC
860 INPUT TAB(15,12)"Nome del file : "ifile$ ''ifile$="D."+file$
870 PROCstoptemp
880 X=OPENOUT ifile$:PRINT#X,ultpassma<:FOR A=0 TO ultpassma<:FOR B=0 TO
    Numservo%:PRINT#X ,R%(A,B):NEXT:NEXT
890 CLOSE #X
900 PROCavviatemp
910 ENDPROC
920
930
940 PROCcarfile: CLS
950 INPUT TAB(15,10)"Nome del file da caricare :"ifile$:ifile$="D."
    +file$
960 PROCstoptemp
970 X=OPENIN file$ :INPUT#X, passo:ultpassma<=passo:FOR A=0 TO passo:
    FOR B=0 TO N Numservo%:INPUT#X ,R%(A,B) :NEXT:NEXT
980 vpasso=0
990 CLOSE#X
1000 PROCavviatemp
1010 ENDPROC
1020
1030
1040 PROCprepara
1050 passo=0 : Numservo%=3: REM NUM. DEI MOTORI-1
1060 ultpassma<=0 :vpasso=0 :rit%=10
1070 passomax=100
1080 DIM R%(passomax,Numservo%),npos% 8 ,temp% 12 , leggi% 12
1090 PROCsommatempo
1100 ENDPROC
1110
1120
1130 DEF PROCsommatempo
1140 REM ********************************
1150 REM Prepara il temporizzatore ecc.
1160 REM ********************************
1170 osbyte=&FFF4
1180 A%=&97 :X%=&62 :Y%=&FF
1190 CALLosbyte: REM predispone la porta B per l'uscita
1200 DIM p%(8)
1210 DIM temp% 12 ,leggi% 12
1220 xtemp=temp% MOD 256
1230 ytemp=temp% DIV 256
1240 xleggi=leggi% MOD 256
1250 yleggi=leggi% DIV 256
1260 PROCinizializza
1270 FOR I%=angolo TO angolo+8:angolo?I%=128:NEXT
1280 T=.02  : REM intervallo in sec. tra impulsi successivi
1290 tempo%=&FFFFFFFF -(t*100) +1
1300 temp%/?4=&FF   : REM carica il byte piu' alto
1310 !temp%=tempo% : REM prepara il temporizzatore, abilita gli eventi
1320 PROCavviatemp
1330 ENDPROC
1340
1350 DEF PROCavviatemp
1360 *FX14,5
1370 A%=4 :X%=xtemp  :Y%=ytemp  :CALL &FFF1
1380 ENDPROC
1390
1400 DEF PROCstoptemp
1410 *FX13,5
1420 ENDPROC
1430
1440
1450 DEF PROCinizializza
1460 REM ********************************
1470 REM Assembla il cod. macchina
1480 REM ********************************
1490 DIM spazio% 500
1500 FOR C=0 TO 3 STEP 3
1510 pagzero=&70 : REM a disposizione dell'utente
1520 portab=&FE60 :osword=&FFF1
1530 P%=spazio%
1540 angolo=P%  :P%=P%+8   : REM fino a 8 motori
1550 tabella=P% :P%=P%+256 : REM 256 durata massima dell'impulso
1560 FOR I%=tabella TO tabella+&100:?I%=&FF:NEXT
1570 tabbassa%=tabella MOD 256
1580 tabalta%=tabella DIV 256
1590 ?pagzero=tabbassa% :pagzero?1=tabalta%
1600 [OPT C
1610 .gestoreeventi
1620 PHP:PHA:TYA:PHA:TXA:PHA
1630 LDA #&04
1640 LDX #xtemp
1650 LDY #ytemp
1660 JSR osword
1670 LDY #&0
1680  \Trasmette l'impulso,  con alcuni motori e'  possibile
1690  \farlo partire prima di caricare la tabella: ridurre
      \il ciclo di pausa qui sotto
1700 LDA #&FF :STA portab
1710  \carica le eccezioni in tabella
1720 LDX #&7 :LDA #&FF  :CLC \prepara una configurazione di bit
1730 .eccezioni
1740       ROR A : PHA  \configurazione di bit
1750       LDY angolo,X \estrae l'offset  corrispondente  all'angolo
               del motore X
1760       AND (pagzero),Y \conserva la configurazione corrente
1770       STA (pagzero),Y \ma modifica quella per il motore x
1780       PLA :DEX
1790       BPL eccezioni
1800  \la tabella e' pronta, piccola pausa
1810  LDY #&60
1820 .attende DEY
1830          BNE attende
1840 LDA #&FF  \tutti gli impulsi attivi...
1850 LDY #&0
1860 .ciclo  AND (pagzero),Y \vengono mascherati
1870         STA portab  \con ciascun elemento della tabella
1880         INY
1890         BNE ciclo
1900 LDX #&7  :LDA #&FF
1910 . cancella
1920   LDY angolo, X \rimuove tutte le eccezioni
1930   STA (pagzero),Y
1940 DEX
1950 BPL cancella
1960  \gli impulsi dovrebbero essere terminati
1970 PLA:TAX:PLA:TAY:PLA:PLP
1980 RTS
1990
2000
2010 .azionaservom  \...  in cod.  macchina il movimento  e' migliore...
2020 .Ciclo 2
2030 LDX#0
2040 LDX#0
2050 .Ciclo 1
2060  LDA npos%,X          \la nuova posizione del servom. x
2070  CMP angolo,X         \e' identica alla posizione precedente?
2080  BEQ mosso            \se uguale, nessuna azione
2090     BCS somma         \se maggiore (), somma
2100        DEC angolo,X   \altrimenti sottrae
2110        JMP incrementa<
2120      .somma
2130        INC angolo,X
2140        JMP incrementa<
2150      .mosso
2160          INY
2170    .incrementa<
2180    INY  :CPX #4
2190  BNE Ciclo 1       \ha verificato i 4 servomotori?
2200  JSR pausamotori
2210  CPY #4
2220  BNE Ciclo 2       \continua se l'operazione non e' conclusa
2230  RTS               \tutti i motori in posizione
2240
2250
2260 .pausamotori
2270 TXA: PHA:TYA:PHA
2280 LDY #&FF :LDX &81
2290 .attesa
2300  DEY :NOP :NOP
2310  BNE attesa \loop di 8 cicli di clock  (per  es.  1024
                 cicli in tutto)
2320 LDY #&FF
2330 DEX
2340 BNE ATTESA
2350 PLA:TAY:PLA:TAX
2360 RTS
2370
2380 ]
2390 NEXT
2400 !&220=gestoreeventi OR (!&220 AND &FFFF0000)
2410 ENDPROC
```

Linguistica strutturale

I concetti di "scopo" e "passaggio di parametri" nelle procedure del PASCAL

Prima di completare lo studio dell'organizzazione dei dati in PASCAL, vengono esaminati alcuni meccanismi, relativi alla strutturazione dei programmi, che permettono di creare procedure e funzioni da aggiungere a quelle proprie del PASCAL. Tuttavia, prima di passare alla realizzazione di nuove procedure, è bene sottolineare brevemente le differenze tra una procedura e una funzione, riesaminando la procedura write e la funzione sqrt. Quando si scrive:

```
write ('Salve!')
```

ci aspettiamo che il parametro stringa di write venga trascritto nel file di uscita. In altre parole, write identifica un sottoprogramma per l'invio di un'informazione all'unità di uscita (VDU) sotto forma di flusso di caratteri. Il seguente comando:

```
sqrt (256)
```

produce invece un messaggio d'errore durante la compilazione, poiché l'istruzione non è formulata correttamente, mentre:

```
write(sqrt(256))
```

è una procedura consentita, che visualizza sullo schermo il valore 1.60000E+01. Riflettendo su questi semplici esempi, è facile comprendere la differenza fondamentale che passa tra una procedura e una funzione.
L'identificatore write avvia un procedimento che trasmette un dato ad un flusso in uscita ed è una procedura che può essere chiamata per nome. Un'altra procedura del PASCAL è page (F) che viene usata per creare una nuova pagina sul file di testo F. L'identificatore sqrt, al contrario, deve essere sempre seguito da un solo argomento numerico e produce come risultato un numero reale, cioè la radice quadrata del suo argomento. La prima differenza tra le procedure e le funzioni è che queste ultime non possono esistere come istruzioni indipendenti: espressioni come sqrt (X/3) o odd (N), a sé stanti, non hanno dunque alcun significato. Gli identificatori di funzione possono essere paragonati a variabili non inizializzate, il cui valore viene elaborato ogni volta che il loro nome compare in una linea di programma. Il risultato dipende, ovviamente, dal valore specificato nell'argomento della funzione. In altre parole, il risultato è una "funzione" dell'argomento.
Le procedure, viceversa, non producono alcun valore e non possono essere usate all'interno di espressioni. Quindi l'istruzione:

```
N := WriteLn
```

è palesemente errata, quanto può esserlo, in BASIC, il comando LET N = PRINT. La creazione di vasti programmi in PASCAL viene facilitata dalla possibilità di attribuire nomi propri, sia alle procedure sia alle funzioni, e di controllarne l'accesso e le reciproche connessioni. Come già visto per i nomi delle variabili, la sostanziale assenza di restrizioni relative agli identificatori consente al programmatore di assegnare nomi semplici ed esplicativi ai sottoprogrammi. Si prenda ad esempio il seguente programma:

```
PROGRAM Incompleto (input,output,filedati);
                    dichiarazioni ...
BEGIN
    Apri (filedati);
    WHILE NOT EoF (filedati) DO
        BEGIN
            read (filedati, elemento);
            Elabora (elemento)
        END
END.
```

Anche se non si è ancora affrontata la gestione dei file in PASCAL, non ci dovrebbero essere problemi nel comprendere il funzionamento del programma. In questo caso si dà per scontato che la sezione dichiarativa contenga la definizione di due procedure: una per individuare un file di dati nella memoria di massa, aprendolo "in lettura" (Apri) e l'altra per elaborare in modo appropriato i diversi elementi dell'informazione (Elabora). La procedura read, finora usata soltanto per leggere caratteri immessi da tastiera, può essere adoperata anche per leggere qualsiasi dato, strutturato o no, da un file di tipo adeguato. La funzione Booleana EoF fornisce una risposta "vera" quando è stato letto l'ultimo record del file assegnatole come argomento. Se l'identificatore del file viene omesso, la funzione si riferisce automaticamente al flusso d'immissione standard. Relativamente ai file di testo, la funzione EoLn (End of Line) fornisce una risposta "vera" non appena è stato letto l'ultimo carattere di una linea. Vediamo adesso come il programmatore PASCAL può usare queste funzioni per realizzare sottoprogrammi.
In sostanza, le differenze tra programmi, procedure e funzioni sono minime: tutti questi moduli possono avere proprie descrizioni "locali" di dati e corpi di istruzioni: cambia solo la sintassi per il titolo. Come già visto, il titolo di un programma è composto dalla parola riservata PROGRAM, da un nome definito dall'utente e da una lista di parametri che indica i file con cui il programma comunica.

Il titolo di una procedura sostituisce la parola riservata con PROCEDURE e amplia l'elenco formale dei parametri, estendendolo agli identificatori di tipo per ciascun parametro dei dati. Ad esempio:

```
PROCEDURA Linee (NumLinee :integer);
    VAR
        N : integer;
    FOR N := 1 TO NumLinee DO
            WriteLn
END;
```

L'istruzione END della procedura è seguita da un punto e virgola, e non da un punto, come accade invece per la END a fine di un programma. NumLinee, l'identificatore formale del parametro, riceve il proprio valore nell'istruzione di chiamata:

```
Linee (5)
```

Questa semplice procedura si dimostra molto utile quando si desiderano inserire più volte numerose linee bianche (o "vuote") in un testo, evitando di scrivere, ogni volta, una noiosa serie d'istruzioni WriteLn separate da punti e virgola. Inoltre la procedura consente di inserire un numero qualsiasi di linee: quindi, Linee (10) produce 10 linee vuote, Linee (20) ne produce 20 e così via.

Scopo

Il seguente esempio fornisce una chiara idea dell'affidabilità del PASCAL. La variabile FOR per il controllo del ciclo iterativo deve sempre essere dichiarata come una variabile reale locale. Consideriamo ad esempio l'istruzione:

```
For Nun Lines := 1 TO NumLinee DO...
```

Sebbene questa linea possa comparire all'interno di un programma principale, non è possibile garantire l'affidabilità del ciclo se la variabile di controllo del ciclo risulta nonlocale o "relativamente globale" (per maggiori informazioni sulle variabili "locali" e "globali", consultare le pagine 1073 e 1168). Nel debugging di un programma BASIC, contenente le seguenti istruzioni:

```
300 FOR N = 1 TO T
400 GOSUB 2000
500 NEXT N
```

può accadere che una subroutine, magari chiamata in modo indiretto o condizionale, usi la variabile N per altri scopi, introducendo una potenziale "bomba" nel programma. Il PASCAL impedisce il verificarsi di una simile situazione, facendo risparmiare ore e ore di lavoro: ad ogni identificatore dichiarato all'interno di un blocco viene assegnata una specifica regione del blocco, di sua esclusiva competenza. Tuttavia il suo "scopo" (ossia quella parte di blocco che ne delimita l'accesso) si estende solo dal punto della sua dichiarazione fino al termine di quel blocco e può essere ulteriormente limitato con una successiva ridefinizione. Nell'esempio precedente, la N assegnata nella procedura Linee è, ovviamente, una variabile intera di tipo locale la cui dichiarazione prevale temporaneamente su qualsiasi altra dichiarazione globale relativa ad ogni attivazione della procedura.

Esaminando le linee generali del programma *Scope* (riquadro in alto), si nota che la regione di pertinenza delle variabili globali N e X comprende l'intero programma. Lo scopo di X inizia dalla sua dichiarazione e termina alla fine del programma, con esclusione della procedura A. Infatti, al suo interno, viene definita una seconda X (SET OF char), quale identificatore di tipo, la cui regione comprende il blocco di A. Analogamente, all'interno della procedura B, N fa riferimento al parametro formale di B e non alla variabile globale. Le regioni di A e di B comprendono l'intero programma, ma solo lo scopo di B inizia dalla sua definizione: è quindi possibile usare A all'interno della procedura B, mentre B risulta "invisibile" ad A. Questa affermazione è un'ulteriore dimostrazione della incisiva logica del PASCAL: nessun programma può essere eseguito finché non è stato completato! Inoltre chiarisce perché le definizioni e le dichiarazioni devono essere disposte nella sequenza CONST, TYPE, VAR ed essere seguite dalle definizioni di procedura e di funzione ordinate in base alle necessità del programma. Molti compilatori PASCAL sono a "passaggio unico" (cioè leggono il codice sorgente una sola volta) e non potrebbero funzionare se l'ordinamento logico del programma fosse diverso da quello appena descritto!

I vantaggi di questa strutturazione ripagano ampiamente l'impegno profuso nella dichiarazione dei dati locali di ogni procedura. La modularità viene ottenuta trasferendo alle procedure tutti i valori dei dati sotto forma di parametri. Sebbene esistano programmi PASCAL che adoperano procedure senza alcuna lista di parametri, accedendo a tutti i dati in modo globale, tale impostazione andrebbe

C'È SCOPO E... "SCOPO"

```
PROGRAM Scopo;

VAR
    N :intero;
    X :reale;
PROCEDURE A (Y : reale);
    TYPE
      X=SET OF char;
    [ecc.]

PROCEDURE B (N : intero);
    [ecc.]

BEGIN [Programma principale - Scopo]
    ...
    A (succ(N)/3);
    B (N);
    A (X)
        [ecc.]
END.
```

La seguente tabella mostra le regioni e gli scopi delle diverse variabili e procedure contenute nel programma "Scopo", qui a fianco:

Proc/Var	Regione	Scopo
A	principale	A, B, principale
B	principale	B, principale
N (globale)	principale	A, principale
X (reale)	principale	B, principale
X (proc. A)	A	A
N (proc. B)	B	B

evitata. Si noti che, così come ogni riferimento a X all'interno della procedura A implica il tipo X, l'uso di N all'interno di B rimanda al suo parametro formale locale, mentre la N intera globale risulta temporaneamente inaccessibile. Oltre a garantire una protezione intrinseca, ciò consente a più programmatori di lavorare allo sviluppo di un programma di grosse dimensioni senza doversi preoccupare di eventuali interferenze tra identificatori.
La chiamata di B passa il valore di N globale, come un parametro reale, ad una variabile, interna a B, che può avere lo stesso nome ma risulta totalmente indipendente. Questo trasferimento implica alcune importanti considerazioni:

- All'ingresso del blocco viene eseguita una copia locale del valore fornito.
- Qualunque cambiamento di questo "parametro di valore" all'interno del suo blocco non modifica il parametro reale fornito dal punto di chiamata.
- Il valore trasferito può essere una qualsiasi espressione, purché di tipo corretto.

Naturalmente ogni chiamata a una procedura deve possedere una lista di parametri reali, che devono corrispondere in numero, posizione e tipo alla lista di parametri formali contenuta nella definizione della stessa procedura. Il meccanismo di trasferimento del valore del parametro può essere interpretato come l'inizializzazione dell'identificatore del parametro formale al valore specificato nell'istruzione di chiamata. Quindi:

Linee (succ(N + spazio)DIV 2)

implica la seguente istruzione di assegnazione relativa al valore di Linee:

NumLinee := succ(N + spazio)DIV 2

Tenendo in mente questa tecnica di trasferimento "protetto" dei valori, si provi a scrivere un'unica procedura capace di elaborare dati i cui valori cambiano di volta in volta.

CONVERSIONE DI NOTAZIONE

Il programma "ValoreBase" chiarisce l'uso delle procedure dotate di parametri di valore e consente di selezionare qualsiasi "base" o notazione compresa tra 2 (base binaria) e 16 (base esadecimale): i numeri decimali inseriti da tastiera vengono visualizzati nella notazione desiderata. Per esempio, il numero 32767, ossia il "massimo intero" (MaxInt) su molti piccoli compilatori PASCAL, viene trasformato nell'esadecimale 7FFF, nel binario 111111111111111 oppure nell'ottale 77777. Si provi a introdurre le seguenti modifiche:

- Per trattare i valori negativi, è necessario convertire il numero nel suo "complemento a 2". A livello macchina, questa operazione viene eseguita facendolo precedere da un segno negativo e sommando 1. Come è possibile ottenere lo stesso risultato in PASCAL?
- Si provi adesso a effettuare le conversioni opposte, usando il programma per trasformare un numero espresso in qualsiasi base (da 2 a 16) in un valore decimale. È possibile inserire questa versione modificata nel programma precedente sotto forma di procedura?

Un piccolo suggerimento: si rifletta sui dati e sulla concatenazione

```
PROGRAM     ValoreBase    (input, output);
        ( converte  i numeri decimali in valori  con  basi
          da binaria a esadecimale)
CONST
        Colonne   = 79;  (per schermo/stampante)
TYPE
        byte      = 0 .. 255;
        cardinale = 0 .. MaxInt;
VAR
        numero,
        base      : intero;
        Stufo,
        legale    : booleano;

        {!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!}

PROCEDURE   ScriviCifra   (cifra : byte);
    {valore della lista [calcolo] fornito alla cifra}
  BEGIN
    IF cifra IN [0 .. 9]
      THEN
        write (cifra : 1)
      ELSE      {rappresenta come A .. F}
        CASE cifra OF
          10     : write ('A');
          11     : write ('B');
          12     : write ('C');
          13     : write ('D');
          14     : write ('E');
          15     : write ('F')
        END      {CASE}

  END;  {ScriviCifra}
        {!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!}

PROCEDURE   Stampa       (N     : cardinale;
                          base  : byte);
  CONST {tutti questi dati sono locali per Stampa}
        CifraMax = 32;
  TYPE
        limiti   = 1 .. CifraMax;
  VAR
        lista    : ARRAY [limiti] OF byte;
        indice   : limiti;
        calcolo  : byte;

BEGIN
  calcolo :=0;

  REPEAT
      calcolo := succ (calcolo);
      lista [calcolo] := N MOD base;
      N := N DIV base

  UNTIL N = 0;
              {stampa iniziando con MSB :}
  FOR calcolo := calcolo DOWNTO 1 DO
      ScriviCifra (lista [calcolo])

  END;  {Stampa}
        {!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!}
BEGIN   {ValoreBase - Programma principale}

  REPEAT
    WriteLn ('Scegli una base per il numero : ');
    WriteLn ('       2 .. 16        ' : Colonne);
    WriteLn ('(altri valori per uscire)' : Colonne);
    write ('Base ? ');
    read (base);
    legale := base IN [2 .. 16];

    IF legale THEN
      BEGIN
        write ('Numero (0 = cambio base) ? ');
        read (numero);
        Stufo := numero <= 0;

        WHILE NOT Stufo DO
          BEGIN
            write (numero : Colonne DIV 2,
              ' in base ', base : 1, ' vale ');
            Stampa (numero, base);
            WriteLn;
            write ('Numero ? ');
            read (numero);
            Stufo := numero <= 0;
          END
      END

  UNTIL NOT legale

END.
```

Le classi del futuro

Come la tecnologia del computer può influire sulla scuola del domani

Nel prossimo futuro i sistemi didattici computerizzati trasformeranno radicalmente l'attuale sistema educativo. Il calo nel costo dell'hardware, il perfezionamento del software e lo sviluppo tecnologico stanno modificando profondamente la scuola: l'istruzione sta rapidamente divenendo uno dei settori d'avanguardia nello sviluppo di una nuova tecnologia. Il motivo di questo salto di qualità dell'istruzione computerizzata è abbastanza semplice. Nei primi anni della "rivoluzione" informatica, le case produttrici di hardware e software hanno puntato su prodotti più remunerativi di quelli didattici. La situazione è rapidamente cambiata quando i fabbricanti hanno compreso l'importanza del mercato scolastico, che permetteva di far conoscere il proprio nome ai ragazzi, potenziali futuri acquirenti dei loro prodotti. Il primo risultato di questa politica commerciale è stato un calo nel costo dell'hardware e una crescita della gamma di apparecchi sul mercato. Nel 1985 la Apple ha promosso una campagna di vendita nelle scuole inglesi con sconti del 30%, mentre la Atari ha lanciato una nuova serie di micro a 32 bit, progettata tenendo presenti le esigenze del mercato didattico, e offre apparecchi dotati di linguaggio LOGO residente in ROM in sostituzione del BASIC. In Gran Bretagna si è verificato un vero e proprio "boom": il mercato scolastico è freneticamente conteso da quattro compagnie (la Amstrad, la Acorn, la Sinclair e la RML). La Amstrad, addirittura, si pone come obiettivo di vendere alle scuole ben 100 apparecchi al giorno!



ALAN ADLER

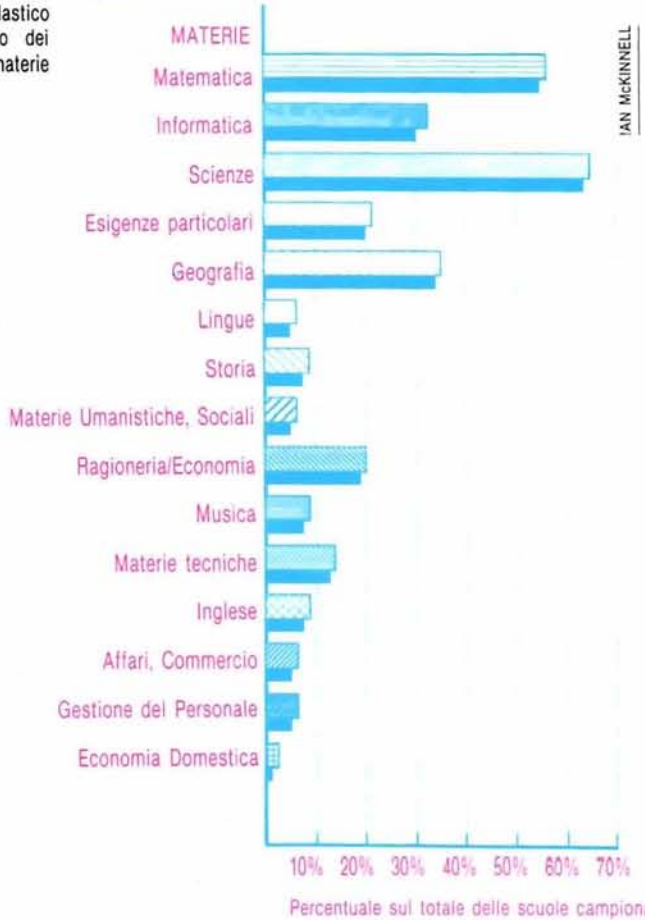
Come studieremo domani

I computer possono rivoluzionare l'attuale sistema d'insegnamento, ponendo l'allievo al centro del processo d'apprendimento e consentendogli di controllare direttamente i diversi sussidi didattici offerti dalle nuove tecnologie. Le comunicazioni via satellite e i sistemi in rete permettono allo studente di collegarsi alle banche dati dei cinque continenti. Le scuole sono dunque destinate a divenire "centri di supervisione didattica", dove i ragazzi si ritrovano per discutere del proprio corso di studi, gran parte del quale può essere svolto a casa. I sistemi a "video interattivo" possono costituire la base per la realizzazione di software più complesso, mentre i robot possono essere usati per la dimostrazione fisica di concetti astratti. Inoltre gli studenti possono collegarsi a sistemi in rete locali, usando un micro come terminale "intelligente", le cui unità centrali gestiscono lo scambio di informazioni, le comunicazioni tra studenti e offrono, quando è necessario, una maggior potenza d'elaborazione

Oltre all'ingresso dell'informatica nella scuola, si è formata una seconda generazione di utenti che desiderano adoperare i propri computer per scopi più professionali. Alle case di software, che in precedenza si erano dedicate essenzialmente allo sviluppo di pacchetti educativi, sta quindi aprendosi un nuovo mercato altamente remunerativo. Alla luce di questa situazione in continua evoluzione, l'uso dei computer in campo didattico può oggi procedere in due direzioni: l'estensione e il perfezionamento dei metodi e l'introduzione di tecniche del tutto nuove.
Seymour Papert ha già indicato una varietà di possibili sviluppi dei concetti base del LOGO, consentiti dalle notevoli capacità di apparecchi quali il Macintosh. Papert ipotizza la creazione di un "microcosmo fisico", realizzato in questo linguaggio, che consenta al bambino di giocare con le leggi del moto, così come accadeva con quelle della geometria. Alcune delle idee di Papert concernenti questo microcosmo sono già state incorporate nella versione del LOGO per il Macintosh. Le simulazioni, i programmi CAL e i database possono diventare sempre più potenti, ma anche più facili da usare, se programmi quali il *Filevision* per Macintosh vengono impiegati nelle scuole.
I settori più stimolanti dell'informatica didattica riguardano tuttavia l'introduzione di tecnologie completamente nuove, in particolare quelle basate sulle CD-ROM (memorie di sola lettura realizzate su "compact disc"), e i "video interattivi" (IV). La possibilità di immagazzinare grandi quantità di dati a costi ragionevoli permette al software didattico di superare l'impostazione "lineare" tipica dei programmi sviluppati su sistemi piccoli e di non facile approccio. Un programma CAL che occupi 48 Kbyte possiede un grado di interazione estremamente limitato e impone notevoli restrizioni al processo di apprendimento del bambino. La tecnologia IV permette di sviluppare programmi che offrono allo studente un approccio più elastico alla materia di studio e mette a disposizione una maggior quantità di memoria, un più agevole movimento all'interno del programma.
Durante un convegno dedicato alle tecnologie didattiche, sono state individuate quattro aree particolarmente adatte all'uso dei video interattivi nelle scuole. Gli insegnanti possono usarli come "sussidi didattici" per la rapida illustrazione di grafici o di sequenze d'immagini, in abbinamento a programmi d'istruzione e a sistemi audiovisivi. Gli studenti possono adoperarli in piccoli gruppi (come "istruttori" per approfondire un particolare argomento e rispondere alle loro domande) oppure per l'apprendimento individuale e per eseguire simulazioni interattive o sofisticati programmi CAL. Infine possono essere impiegati per accedere a informazioni, come ad una biblioteca di consultazione audio e video.
Sebbene la tecnologia IV implichi una note-

I video interattivi

Le applicazioni scientifiche hanno finora rappresentato il principale obiettivo dell'introduzione dei microcomputer nella scuola. L'introduzione di sistemi dotati di video interattivi e lo sviluppo di software per IV possono contribuire notevolmente ad un uso didattico del computer molto più generalizzato. L'insegnamento della storia, ad esempio, può essere radicalmente trasformato dall'impiego di simulazioni complesse (ma facili da usare), grazie alle immense capacità d'archiviazione di dati offerte dalla tecnologia dei video dischi. Qui di fianco è riassunto in un grafico il risultato di una ricerca, compiuta nel settore scolastico d'oltremanica, sull'utilizzo dei computer suddiviso per materie scolastiche



IL DOMESDAY PLUS 900

La enorme capacità di memoria dei video dischi offre possibilità di impiego,completamente nuove. Per commemorare il novecentesimo anniversario del "Domesday Book" (il Libro del Catasto compilato per ordine del re Guglielmo I dal 1086 in poi), la BBC, la Philips, la Thorn-EMI e il Microcomputers in Education Project hanno lanciato un progetto su scala nazionale per la raccolta di dati sull'intero territorio britannico.
Le scuole di tutta la nazione raccoglieranno notizie relative al territorio in cui si trovano, che verranno poi inserite in un database accessibile al pubblico. Alcune informazioni verteranno sulle attrazioni paesaggistiche, sui servizi, ecc. Altre notizie, rilevate da compagnie specializzate, andranno invece a costituire un database su scala nazionale. Entrambi i database, comunque, saranno contenuti in un singolo disco laser.
Il Domesday Project offrirà un quadro aggiornato della odierna società inglese e grazie alla tecnologia dei dischi laser, conterrà testi, mappe, diagrammi, tabelle ed anche sequenze animate



vole mole di lavoro per l'indirizzamento e il recupero delle informazioni e per lo sviluppo del software, il sistema è già stato introdotto a differenti livelli scolastici. La Open University lo ha adottato per i propri corsi estivi durante i quali, per la ristrettezza di tempo disponibile, gli studenti incontrano difficoltà; quelli di ingegneria, ad esempio, nello studio di materie quali l'effetto dell'usura su vari materiali. Il particolare programma per IV qui impiegato, basato su immagini animate e sequenze televisive, riproduce le fasi di un procedimento legale tra i fabbricanti di orsacchiotti di pezza e quelli di rondelle usate per fissare gli occhi al giocattolo. Lo studente deve condurre una serie di esperimenti i cui risultati vengono usati come prove nel giudizio. Il programma interattivo è controllato da un ampio programma CAL, che analizza le possibili cause dello sfaldamento del materiale plastico usato per gli occhi dell'orsacchiotto. Lo studente esegue e interpreta gli esperimenti che vengono poi presentati nella sequenza finale del programma per IV.

Niente più scuole

Il trasferimento delle attuali tecniche software (simulazione, CAL, database, ecc.) sui sistemi a CD-ROM apre interessanti prospettive, ma i futuri effetti di questa nuova tecnologia vanno ben oltre. Papert ritiene che entro il 2000 le scuole scompariranno perché è impensabile che, nell'èra del computer, possano sopravvivere le figure dell'insegnante seduto alla cattedra e dello studente che prende appunti: i ragazzi continueranno a incontrarsi, a studiare e a sviluppare le proprie capacità, ma che il termine "scuola" rimanga o scompaia è irrilevante. Ciò che conta è che la scuola così com'è oggi non esisterà più!
Questa ipotesi viene sostenuta anche dal professor Tom Stonier, secondo cui l'avvento del computer causerà il passaggio dall'attuale istruzione scolastica a quella impartita a casa. Certamente anche il ruolo dell'insegnante subirà un radicale mutamento.
Le possibili implicazioni della transizione da un sistema educativo generalizzato ad uno orientato al singolo studente sono enormi e non si limitano al solo processo d'apprendimento. Stonier è coautore del libro *Children, Computers and Communications* (Bambini, Computer e Comunicazioni) in cui viene ipotizzata un'istruzione basata sul sistema chiamato della "nonna elettronica".
Poiché, sempre secondo Stoner, l'insegnamento è destinato a divenire essenzialmente "domestico" e i progressi della medicina e dei sistemi di salvaguardia della salute accrescono costantemente la durata della vita media, gli anziani torneranno a svolgere quel ruolo fondamentale nell'educazione dei giovani che già avevano nelle società "primitive". Stonier ritiene che gli anziani, oggi spesso relegati ai margini della società, siano particolarmente qualificati per svolgere un proprio ruolo in campo educativo. Un esempio di come potrebbe articolarsi questo nuovo sistema didattico esiste già negli Stati Uniti. A Portage, nel Wisconsin, un insegnante trascorre mezza giornata alla settimana a casa dei suoi studenti; durante la visita, valuta i progressi fatti dall'allievo e decide, assieme ai genitori dello studente, il lavoro per la settimana successiva.
Sebbene il sistema adottato a Portage sia orientato verso bambini portatori di handicap, è certamente possibile estenderlo anche ad altri gruppi di studenti. In Gran Bretagna, ad esempio, è consentito istruire i ragazzi a casa propria, purché muniti di una specifica autorizzazione dall'autorità scolastica competente. Un numero sempre crescente di genitori si affida a questo sistema di istruzione "non scolarizzata". L'accesso a database pubblici di grandi dimensioni, la potenza di micro corredati di database a CD-ROM e il software interattivo rappresentano per i genitori strumenti didattici essenziali per l'educazione "domestica" dei propri figli.
Con l'aumentare della complessità e degli obiettivi del software didattico, l'insegnante gioca un ruolo sempre minore e sempre più difficile da definire. L'insegnante, inteso come "banca dati semovente", come qualcuno lo ha scherzosamente definito, è quindi destinato a scomparire perché la sua obiettiva limitatezza, in quanto essere umano, rischia sempre più di intralciare l'apprendimento dell'allievo. Stonier ritiene che, dato il continuo sviluppo del software didattico e dei database, tale situazione si ripeterà a tutti i livelli del corso di studi. In futuro gli insegnanti dovranno svolgere essenzialmente il ruolo di "consulenti didattici", esperti nelle tecniche di apprendimento più che nella propria materia, e il loro compito non sarà più insegnare materie specifiche, ma piuttosto insegnare ad imparare e aiutare i ragazzi ad orientarsi nei loro studi.



Centro di produzione OPEN UNIVERSITY - BBC

Testimone oculare

Sequenze filmate e immagini ad alta risoluzione richiedono l'elaborazione e l'archiviazione di grandi quantità d'informazioni. Questi obiettivi, fino a ieri irrealizzabili su sistemi controllati da microcomputer, sono oggi possibili grazie alla tecnologia dei video interattivi. Le due immagini in figura sono state tratte dal programma didattico per studenti d'ingegneria "The Teddy Bear Disc", eseguito su un sistema a video disco controllato da un computer. La tecnologia IV permette allo studente di interpretare il ruolo di "esperto" e di eseguire esperimenti i cui risultati vengono poi usati come prove nel processo che conclude la simulazione

RIVOLUZIONE CD-ROM

Per rendere più affidabile la registrazione dei dati, gran parte della memoria delle CD-ROM è riservata alle routine per la verifica degli errori e alle informazioni di formattazione. Nonostante ciò, un disco CD-ROM offre all'utente ben 552 Megabyte di memoria, sufficienti per immagazzinare migliaia di immagini. Inoltre, a differenza dei giradischi per Compact Disc audio, in cui sono impiegati sofisticati convertitori D/A (digitale/analogico), le piastre CD-ROM trattano solo dati digitali: ciò consente di semplificare i circuiti del giradischi e di ridurre i costi di produzione. Il prezzo di vendita di tali prodotti resta tuttavia legato alla domanda del mercato e alla standardizzazione delle tecniche di fabbricazione.
Poiché un solo disco CD-ROM può facilmente contenere tutti i programmi applicativi necessari ad un'utenza media, la loro comparsa in commercio è destinata a rivoluzionare l'intero mercato del software

Variabili di valore

Alcuni esempi della ferrea e concisa struttura del Pascal

A differenza delle procedure, che vengono definite e chiamate specificandone il nome, le funzioni si possono adoperare soltanto nelle espressioni e non come istruzioni a sé stanti. Mentre le chiamate di procedura implicano l'elaborazione di sottoprogrammi per eseguire un particolare procedimento, le chiamate di funzione calcolano un valore. Il risultato può essere di qualsiasi tipo semplice (reale o scalare) oppure, come si è già visto, un puntatore. La sola differenza nella relativa testata, oltre alla presenza della parola riservata FUNCTION, è l'obbligo di indicare l'identificatore di tipo del risultato ottenuto. Ad esempio, se la versione di PASCAL usata non possiede una propria funzione Odd (dispari), il programmatore può così definirla:

```
FUNCTION Odd (numero : integer) : boolean;
    BEGIN
        Odd := numero MOD 2>0
END; Odd
```

Poiché il risultato viene depositato nell'identificatore della funzione, esso va assegnato all'interno della funzione (si rammenti che nelle assegnazioni il nome della funzione deve comparire nella parte sinistra dell'espressione). I nomi di funzioni, sono paragonabili a variabili non inizializzate e il loro valore viene calcolato ogni volta che compaiono in una espressione. Perciò, se si scrive:

```
IF numero MOD 2>0 THEN
    Odd := true
```

(omettendo la necessaria clausola ELSE Odd:= false), esiste la possibilità che il risultato restituito sia indefinito. Come accade per i parametri di procedura fin qui esaminati, il valore del parametro effettivo viene trasferito dal punto d'attivazione all'identificatore dell'intero locale, ossia numero. Quindi:

```
WriteLn ( Odd ( sqr ( N DIV 100 )))
```

che dovrebbe sempre dare come risultato False, esegue quattro operazioni:

1. calcola l'espressione N DIV 100;
2. trasferisce il valore dell'intero temporaneo a sqr come un effettivo parametro di valore;
3. calcola il risultato di sqr e lo passa a Odd;
4. infine valuta il risultato Booleano e lo fornisce alla procedura WriteLn come un normale parametro di valore.

Se, prima che la precedente frase venga eseguita, N vale 17, quale sarà in seguito il valore di N? Non c'è alcun motivo per cui il valore di N debba cambiare durante la valutazione di una funzione che ha N come parametro. Il valore ottenuto, infatti, è una "funzione", appunto, del valore dei suoi "argomenti" o parametri. Il meccanismo seguito dal PASCAL per fornire solo il valore delle variabili assicura che, anche quando i parametri vengono cambiati localmente da una funzione o da una procedura, i parametri effettivi non vengano alterati al punto di attivazione, poiché in effetti il Pascal opera su delle copie locali di parametri.

Dati globali contro parametri

Per questi stessi motivi, sebbene il PASCAL non lo vieti espressamente, non è possibile accedere ai dati in modo globale. Tutte le concatenazioni di dati tra la procedura e le chiamate di funzione devono essere controllate da tabelle di parametri, anche se i parametri si trovano all'interno dello "scopo" del sottoprogramma. La sola eccezione a questa regola generale riguarda l'accesso alle costanti globali. A causa della loro particolare natura, le costanti globali in PASCAL sono protette contro qualsiasi alterazione: è impossibile modificarne il valore. Tuttavia, se il valore di una costante viene fornito sotto forma di parametro, essa assume il ruolo di una variabile locale e perde ogni protezione.

```
FUNCTION Minuscola (caratt :char) : char;
    produce la minuscola di qualsiasi argomento maiuscolo
    CONST
        scarto = 32; ASCII ord ('a') — ord ('A')
    BEGIN
    IF caratt IN ['A' .. 'Z']
        THEN
            Minuscola := chr(ord (caratt) + scarto)
        ELSE
            Minuscola := caratt
    END; Minuscola
```

Nel definire una procedura, talvolta è necessario che questa modifichi i valori dei propri parametri, altrimenti non potrebbe eseguire il compito per il quale essa viene creata. Un esempio ovvio è la stessa procedura read del PASCAL: read(N) sarebbe un'espressione del tutto inutile se la procedura assegnasse un valore ad un intero locale per la read e il valore di N non venisse modificato. In casi simili, è necessario passare l'indirizzo di un parametro variabile, piuttosto che il suo valore, in modo che la procedura possa riferirsi a questo e non ad una copia locale. Ciò viene detto "passaggio per indirizzo" o "per riferimento" e viene abitualmente usato solo con le procedure (e non con le funzioni).

Per il passaggio di variabili "parametriche" è sufficiente far precedere la parola riservata VAR ai relativi elementi della tabella dei para-

metri della procedura. La sintassi di una lista di parametri è simile alla dichiarazione VAR di un blocco, ma la parola VAR, anziché apparire una sola volta come delimitatore, compare solo prima del parametro che deve essere modificato dalla procedura:

```
PROCEDURE Procedimento (VAR contat :Contab);
```

Sviluppo di un'applicazione

Il funzionamento di questa tecnica può essere chiarito sviluppando un programma che legga dei nomi inseriti da tastiera, a ciascuno dei quali è associata una quota di denaro. Per semplificare le cose, si usa una sola stringa per ciascun nome e si esprime l'ammontare in lire come valore intero. Dopo aver provato il programma, si possono aggiungere altri particolari quali l'indirizzo, il numero telefonico e così via. Lo scopo è quello di produrre un elenco dei nomi in ordine alfabetico ascendente o in base alla quota di denaro. La struttura naturale dei dati sarà quella di una lista di registrazioni con campi di tipo appropriato, che possono essere aggiunti in seguito. Si può trattare ciascun record come un frammento d'informazione e usare uno qualsiasi dei campi definiti come "chiave" per l'ordinamento (vedere a pag. 1604). Per prima cosa si definisce la struttura generale dei dati:

```
CONST
   LungStringa    = 20;
   LungLista      = 50;
TYPE
   Cardinale      = 0.. MaxInt;
   DimStringa     = 1.. LungStringa;
   stringa        = PACKED ARRAY [DimStringa]
                    OF char;
   dati           = RECORD
                       nome : stringa;
altri campi           ammontare : Cardinale;
                    END; {dati}
   limiti         = 1 .. LungLista;
   ListaReg       = ARRAY [limiti] OF dati;
```

Il programma consente l'immissione di 50 nomi e stabilisce (per adesso) un limite massimo di 30 caratteri per ciascun nome: questo limite può essere modificato intervenendo sulle definizioni. Tutte le sottogamme e le strutture sono state espresse in una parte della definizione TYPE per i seguenti motivi:

1. protezione - un indice del tipo limiti ad esempio, non deve mai superare i limiti definiti per la matrice;
2. ricerca degli errori - se, durante il collaudo del programma, si verifica un "range error", il messaggio d'errore facilita l'individuazione del problema;
3. comodità ed efficienza - le variabili locali possono essere di un tipo già presente: si evitano così duplicazioni e si risparmia memoria;
4. necessità - il PASCAL richiede che gli "indicatori" del tipo di parametro siano identificatori e non "letterali" (ad esempio, 0 .. 255).

Le variabili indispensabili e l'algoritmo del programma vengono descritti in seguito: adesso si provi a progettare un algoritmo approssimativo per il programma e a creare una lista delle procedure ritenute necessarie.

NUMERI... A PROVA D'ERRORE

Il programma "LettoreSchede", qui a fianco, impiega una tecnica per manipolare i numeri positivi "a prova d'errore", poiché legge i dati sotto forma di caratteri. La conversione nell'equivalente valore numerico viene eseguita semplicemente sottraendo il codice del carattere da quello della cifra '0'. Ovviamente il metodo funziona in quanto i codici dei caratteri '0'.. '9' sono numerati progressivamente. Si noti che il programma contiene soltanto due variabili globali (al "livello 0"). I dati per l'elaborazione dei numeri e la funzione Valore, che esegue la correlazione tra valore numerico e carattere, vengono usati solo da "LeggeScheda", quindi vengono dichiarati e definiti all'interno di questa procedura. Il programma ha un solo difetto: se viene immesso un valore superiore al massimo previsto per gli interi del compilatore (MaxInt), l'esecuzione del programma si blocca all'istruzione di assegnazione posta all'interno del ciclo WHILE. Come esercizio, si provi a scrivere una procedura LeggiInt capace di trattare i numeri negativi

```
PROGRAM          LettoreSchede   (input, output);

TYPE
                 cardinale              = 0.. MaxInt;

VAR
                 NumPos  : cardinale;
                 bene    : boolean;

                 {11111111111111111111111111111111}

PROCEDURE        LeggiScheda  ( VAR   N      :cardinale;
                                VAR   OK     :boolean );
   CONST
                 spazio     = ' ';
      TYPE
                 singolo    = 0.. 9;
   VAR
                 simbolo    : char;
                 cifre      : SET OF char;
                   {22222222222222222222}
                     {livello 2}
   FUNCTION Valore ( cifra : char ) : singolo;
                 {calcola il valore numerico di una
                 cifra per qualsiasi set di caratteri}
      BEGIN
         Valore := ord ( cifra ) - ord ( '0' )

      END;       {Valore}
                   {22222222222222222222222}
                 {torna a livello 1}
   BEGIN {LeggiScheda}

      REPEAT
         read (simbolo)

      UNTIL simbolo > spazio;

      cifre := ['0' .. '9'];

      OK := simbolo in cifre;

      IF OK THEN
         BEGIN
            N := 0;

            WHILE simbolo IN cifre DO
               BEGIN {calcola in base 10}
                  N := 10 * N + Valore (simbolo);
                  read (simbolo)
               END {viene scartato qualsiasi carattere di
         END         separazione o 'EoLn' dopo il numero}

   END;       {LeggiScheda}
              {11111111111111111111111111111111}
BEGIN         {LettoreSchede - Livello 0 Programma principale}

   page (output);
   WriteLn;
   WriteLn ('Questo programma legge numeri');
   WriteLn ('interi positivi (compresi tra 0 e MaxInt : 1, ');
   WriteLn ('La conclusione del programma è controllata');
   WriteLn ('da un processo d'individuazione degli errori.');
   WriteLn;
   write ('Numero?');
   LeggiScheda (NumPos, bene);

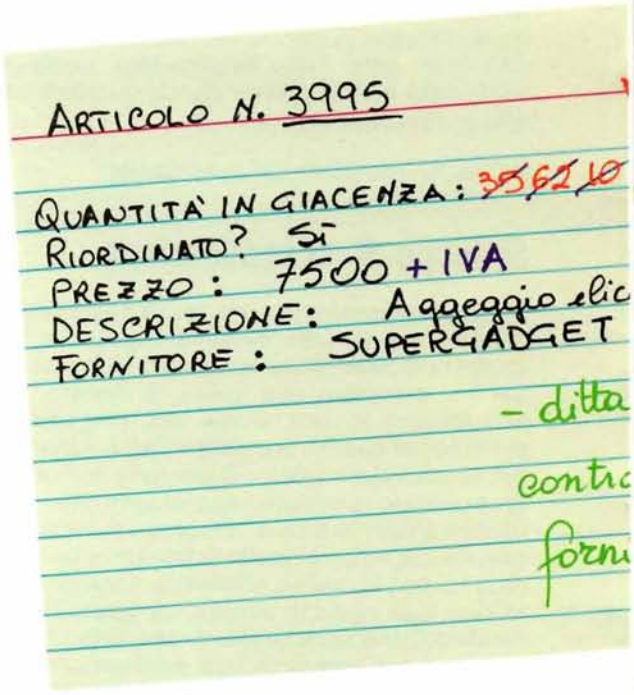
   WHILE bene DO
      BEGIN
         WriteLn;
         write ('Il numero inserito era :');
         WriteLn (NumPos : 1);
         write ('Numero ? ');
         LeggiScheda (NumPos, bene)
      END;

   WriteLn ('--- INDIVIDUATO UN ERRORE ---' : 40)

END.
```

Database a doppio ingresso

I criteri applicativi dei database variano a seconda del tipo



I record che compongono i database cosiddetti "lineari" sono completamente autonomi, quindi ciascun record contiene tutte le informazioni richieste e non è necessario consultarne altri. In questo senso, i tradizionali schedari manuali, come quello di una biblioteca, ad esempio, sono lineari: ogni record può comprendere i campi TITOLO, AUTORE, EDITORE e ISBN (International Standard Book Number). Non è indispensabile alcun riferimento incrociato.

Talvolta, però, un campo contiene un'informazione che può avere una propria specifica registrazione. Ad esempio, nell'archivio dei soci di un club, un campo FIGLI può riferirsi ad altre registrazioni appartenenti allo stesso file (se i figli di un socio sono anch'essi membri del club, vengono registrati anche sotto il proprio nome). Tuttavia un campo può far riferimento a registrazioni che potrebbero non adattarsi alla struttura generale del database. Si consideri il database per un magazzino di pezzi di ricambio, contenente i campi: CODICE DELL'ARTICOLO, PREZZO, QUANTITÀ e FORNITORE. In questo caso è molto probabile che il campo FORNITORE rimandi ad altre registrazioni, che non rientrano nella struttura definita per il database RICAMBI. Per chiarire meglio questi concetti, torniamo all'esempio dell'archivio SOCI DEL CLUB:

NOME	Paolo Bosi
ANNO D'ISCRIZIONE	1979
QUOTA ASSOCIATIVA	Pagata
PROFESSIONE	Insegnante
FIGLI	Luca Bosi, Anna Bosi

I due nominativi abbinati al campo FIGLI potrebbero essere presenti come registrazioni individuali nello stesso file SOCI DEL CLUB, se i figli ne sono membri. Si confronti adesso il record precedente con quello del database RICAMBI:

CODICE ARTICOLO	3995
QUANTITÀ	600
PREZZO	35000
DESCRIZIONE	Carburatore
FORNITORE	Ditta X, Ditta Y

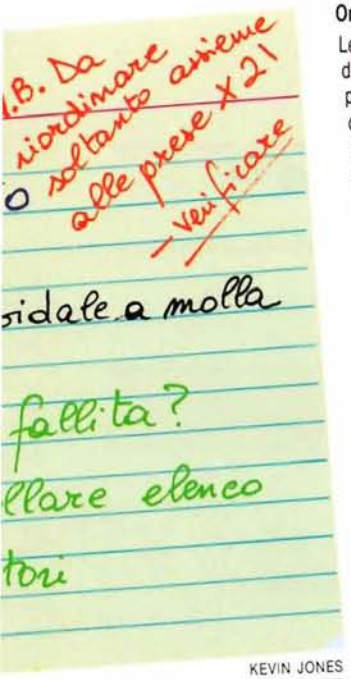
Anche qui c'è un campo, FORNITORE, che contiene due nominativi. Tuttavia la struttura delle informazioni ad essi abbinate non si adatta certo a quella delle altre registrazioni del database RICAMBI. Per completare la conservazione dei dati relativi ai fornitori, dunque, è necessario un secondo database che potrebbe contenere registrazioni del seguente tipo:

FORNITORE	Ditta X
INDIRIZZO	Via Lomellina 139 Milano
TELEFONO	02-793245
PRODOTTO1	Pasta ai siliconi (1/2 kg.)
PREZZO1	6000
PRODOTTO2	Pasta abrasiva (piccola)
PREZZO2	4000
PRODOTTO3	Pasta abrasiva (grande)
PREZZO3	7000
PRODOTTO4	—

La struttura per le informazioni relative a ogni fornitore è sensibilmente diversa da quella per i dati concernenti i RICAMBI. Sono quindi necessari due database distinti: uno per i pezzi di ricambio e l'altro, su un file separato, per ciascuno dei fornitori.

Per ottenere risultati soddisfacenti, è indispensabile possedere un DBM capace di elaborare più file contemporaneamente. I DBM meno sofisticati, quali il *Card-Box* (vedere a pag. 1584), possono gestire un solo file alla volta mentre quelli come il *dBase II* (vedere a pag. 1632) possono accedere simultaneamente a un file primario (per esempio, RICAMBI) e ad uno secondario (ad esempio, FORNITORI). Un DBM di buon livello permette di eseguire un ordinamento su un file primario (per selezionare le registrazioni desiderate) e l'estrazione delle corrispondenti registrazioni (ad esempio, FORNITORI) dai file collegati.

Il pacchetto *dBase II* consente infatti di utilizzare contemporaneamente due file collegati e di associarli usando "campi di chiave" comuni. I due comandi che controllano tali



KEVIN JONES

Ordinazione di magazzino

Le registrazioni previste dal sistema d'archiviazione di uno schedario di propria progettazione possono essere complesse (e forse anche disordinate) quanto si vuole. Nei DBM l'accesso ai dati è necessariamente più rigido e strutturato, ma i programmi più sofisticati permettono di richiamare archivi collegati e, durante l'introduzione dei dati, di saltare specifici campi se questi risultano "doppioni" di altre registrazioni interne al record. Alcuni programmi per database, inoltre, offrono la possibilità di creare campi numerici dipendenti, il cui valore viene automaticamente desunto o calcolato dal contenuto di altri campi

operazioni incrociate sono SELECT PRIMARY e SELECT SECONDARY. Se il file primario e secondario sono rispettivamente RICAMBI e FORNITORE, e il sistema sta lavorando sul file RICAMBI, va impartito il comando USE RICAMBI. Se invece vogliamo eseguire una ricerca incrociata sul file dei fornitori, occorre il comando SELECT SECONDARY seguito, sulla linea successiva, da USE FORNITORE. L'utente può disporre di tutti i normali comandi del *dBase II*, che agiscono sull'ultimo file selezionato.

Talvolta non è detto che il contenuto di un campo (come il campo FORNITORE del file RICAMBI citato in precedenza) richieda l'uso di registrazioni interamente dipendenti (contenuti in file autonomi): in tal caso bastano campi dipendenti, posti all'interno dello stesso record. Per esempio, volendo realizzare un database dedicato alle più recenti tecnologie in campo automobilistico, contenente dati tecnici di vario tipo tratti da libri e riviste specializzate, il database potrebbe avere la seguente struttura:

ARGOMENTO: _____
SINOSSI1: _____
SINOSSI2: _____
SINOSSI3: _____
FONTE: _____ ISBN: _____
TITOLO: _____
DATA: _____
N. PAGINA: _____

Se la fonte d'informazione è una rivista, il campo ISBN (relativo al codice librario internazionale) non va immesso, poiché le riviste non possiedono un simile riferimento, utile invece quando l'informazione proviene da un libro. In entrambi i casi, comunque, viene registrato il numero di pagina, che invece manca qualora le informazioni siano ricavate da un programma televisivo. Alcuni DBM, come il Rescue della Microcomputer Business Systems, consentono di mantenere o eliminare un particolare campo all'interno di un record che dipende (o viene calcolato automaticamente) da altri campi inseriti in precedenza.

Quindi, se la fonte d'informazione non è un libro, il campo ISBN viene eliminato e non compare né sullo schermo né in stampa. D'altra parte, se la notizia non proviene da libri o riviste, il campo N. PAGINA è inutile e non viene visualizzato.

Un DBM con queste caratteristiche permette di risparmiare memoria e di migliorare la rappresentazione visiva dell'informazione, ma è in genere meno versatile dei DBM capaci di correlare le registrazioni contenute in due file diversi.

I database che contengono alcune informazioni di base o "radici" (costanti in tutti i record), e altre "ramificate" (che possono essere presenti o mancare), possiedono una struttura detta "a due livelli". Come indicazione generale, i DBM multi-archivio possono gestire i dati gerarchici a due livelli come sottoinsiemi o casi speciali di un database a più file.

Ad esempio il riferimento per un libro contenuto nel campo FONTE può rinviare al file LIBRI, mentre quello per un programma televisivo può rinviare al file TV.

Un'ultima considerazione. Mentre una persona può progettare e realizzare uno schedario manuale più o meno complesso, i vari DBM attualmente in commercio presentano notevoli limitazioni: prima di acquistare un DBM per il proprio micro, conviene stabilire con precisione l'uso che se ne vuol fare e valutare attentamente le caratteristiche dei pacchetti presi in esame.

DATABASE A CONFRONTO

NOME	CASA	APPARECCHIO	FORMATO	OSSERVAZIONI
Betabase	Clares	BBC	disco	Lunghezza max. record: 2048. N. max campi: 200. Dimensioni max. file: 99K (40T), 199K (80T)
Database	Gemini Marketing	Spectrum	cassetta	Sistema a schedario. Record definiti dall'utente.
DFM Database	Dialog	C64	disco	15 campi/record. Campo alfanum. a 36 caratteri. Campo numerico a 9 cifre.
Practifile	Computer Software Associates	C64	disco	3800 rec/file. Riordina i dati per la corrispondenza. Immissione di tipo "batch"
MicroPen	AMSoft	Amstrad	disco	Spreadsheet e word processor incorporati. Lunghezza max. record: 1024. N. max. record nel file: 32750.

DATABASE

La quarta tabella della mappa di memoria del Commodore 64. Per concessione della Commodore Business Machines

INDIRIZZO ESADECIMALE	ETICHETTA	LOCAZIONE DECIMALE	DESCRIZIONE
0091	STKEY	145	**Flag: STOP key / RVS key** Flag: tasto STOP / tasto RVS
0092	SVXT	146	**Timing Constant for Tape** Costante di temporizzazione per il registratore a nastri
0093	VERCK	147	**Flag: 0 = Load, 1 = Verify**
0094	C3P0	148	**Flag: Serial Bus - Output Char. Buffered** Flag: Bus seriale - Emissione di un carattere dal buffer
0095	BSOUR	149	**Buffered Character for Serial Bus** Deposito per il carattere da emettere sul bus seriale
0096	SYN0	150	**Cassette Sync No.** Numero di sincronizzazione per registratore a nastri
0097		151	**Temp Data Area** Area temporanea per dati
0098	LDTND	152	**No. of Open Files / Index to File Table** Numero di file aperti / Indice alla tabella dei file
0099	DFLTN	153	**Default Input Device (0)** Codice del normale dispositivo di immissione (0)
009A	DFLT0	154	**Default Output (CMD) Device (3)** Codice del normale dispositivo di emissione (3)
009B	PRTY	155	**Tape Character Parity** Parità dei caratteri per il registratore a nastri
009C	DPSW	156	**Flag: Tape Byte-Received** Flag: segnala che un byte proveniente dal registratore è stato ricevuto
009D	MSGFLG	157	**Flag: $80 = Direct Mode, $00 = Program** Flag: $80 = Modo Diretto, $00 = Programma
009E	PTR1	158	**Tape Pass 1 Error Log** Segnale di errore nel 1° passaggio del nastro
009F	PTR2	159	**Tape Pass 2 Error Log** Segnale di errore nel 2° passaggio del nastro
00A0-00A2	TIME	160-162	**Real-Time Jiffy Clock (approx) 1/60 sec** Impulsi jiffy (circa 1/60 di secondo) dell'orologio in tempo reale
00A3-00A4		163-164	**Temp Data Area** Area temporanea per dati
00A5	CNTDN	165	**Cassette Sync Countdown** Contatore all'indietro per sincronizzazione del registratore
00A6	BUFPTN	166	**Pointer: Tape I/O Buffer** Puntatore al buffer di ingresso/uscita del registratore

Il kit Fischertechnik

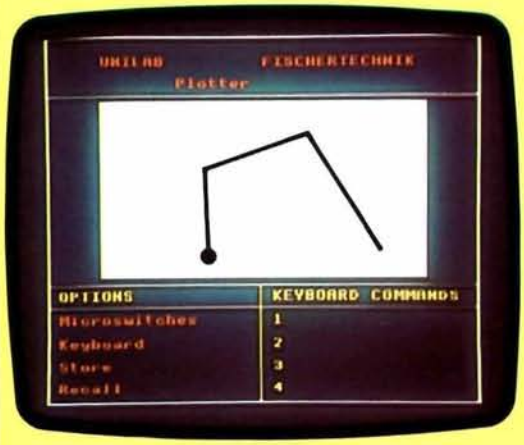
Una scatola di montaggio della Fischertechnik per fare esperimenti con i robot

Nonostante le ottimistiche previsioni di molti operatori del settore informatico, che più volte hanno affermato l'imminenza di una vasta espansione del mercato della robotica "domestica", in realtà questo fenomeno tarda a verificarsi.
La ragione di tale ritardo è semplice: sebbene siano apparsi in commercio svariati "home" robot, il prezzo piuttosto elevato e la parziale inaffidabilità di questi apparecchi hanno scoraggiato l'interesse dei potenziali acquirenti. Da un lato esistono piccoli robot, da costruire, il cui montaggio è semplicissimo e che anche i meno esperti riescono in poche ore a rendere funzionanti, come nel caso dei Movits (vedere a pag. 1564). Il problema è che, proprio a causa della loro semplicità, le applicazioni possibili sono alquanto limitate, facendo smorzare presto gli entusiasmi iniziali. Dall'altro lato il mercato offre robot che, pur appartenendo alla categoria dei "domestici", richiedono nozioni approfondite di elettronica e di robotica. Non sempre, inoltre, tali apparecchi permettono una quantità maggiore di applicazioni. In una situazione come questa, il robot domestico rischia di essere considerato alla stregua di un giocattolo: dopo i fervori iniziali viene relegato in un angolo a prendere polvere.
L'ideale sarebbe un robot in scatola di montaggio, facile da assemblare e da usare, e capace di svolgere più funzioni. È con questo intento che la Fischertechnik ha realizzato la scatola di montaggio Robotics Kit.
Come idea commerciale non è affatto nuova: esistono da anni scatole di montaggio composte da piccoli moduli di facile incastro, che consentono ai ragazzi la costruzione di apparecchi elettronici (radio, amplificatori, ecc.). In qualsiasi momento, i vari moduli possono essere separati e riutilizzati per un nuovo progetto.
Analogamente, intere generazioni sono cresciute giocando con scatole di costruzioni modulari, come quelle prodotte dalla Lego, in cui i pezzi possono essere riutilizzati all'infinito per creare modelli di tutti i generi. La Fischertechnik ha con grande abilità riunito assieme questi due generi di gioco, arricchendoli con le nozioni di robotica, per realizzare un prodotto che può essere senz'altro considerato il primo vero passo verso una diffusione di massa della robotica.
Il Robotics Kit mette in grado di costruire un certo numero di dispositivi robotizzati, alcuni dei quali da pilotare con un comune home computer. Le possibili realizzazioni comprendono un braccio robot, una macchina capace

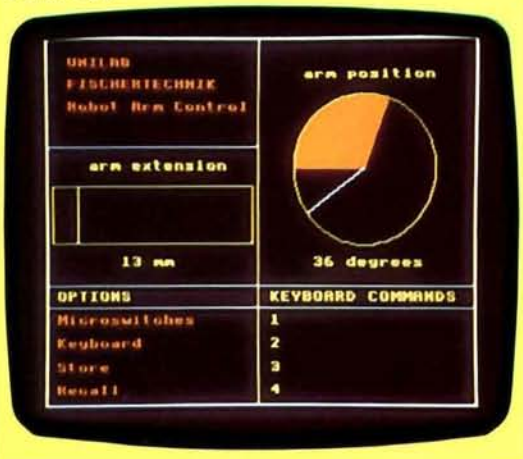
MOVIMENTI SOTTO CONTROLLO

Queste immagini illustrano alcune fasi del funzionamento del software per l'interfaccia Unilab. La foto in basso si riferisce al programma per il controllo del braccio robot: un diagramma (in alto a destra) mostra i movimenti del braccio. Assumendo che la posizione iniziale sia zero, un raggio sul cerchio indica l'angolo di rotazione eseguito. Il diagramma a sinistra, invece, mostra l'estensione orizzontale del braccio. Nella parte inferiore dello schermo compaiono le varie opzioni di controllo offerte dal programma e dalla configurazione degli interruttori sul robot. Esiste la possibilità di memorizzare intere sequenze di movimenti, per rieseguirle a piacere in un secondo momento.
La foto in alto raffigura il funzionamento del programma per il controllo del robot-plotter. Si noti che la sezione dei comandi è identica a quella del programma per il robot, anche se esistono appositi comandi per abbassare o alzare la "penna" del plotter. I movimenti del plotter sono riprodotti nella finestrella dello schermo

Robot-plotter



Braccio robot



di riordinare oggetti di varia lunghezza, un plotter e un dispositivo di immissione per la grafica. Una volta esaurito l'interesse per la sperimentazione con uno dei dispositivi, si può smontare il tutto e passare ad un'altra applicazione. Per poter controllare il dispositivo tramite un computer è necessaria un'adeguata interfaccia (non fornita con il kit), che converta i segnali digitali del computer in quelli necessari per far funzionare i motori del dispositivo.

I componenti della scatola di montaggio, tutti in plastica o perlomeno rivestiti in plastica, si incastrano tra loro in modo simile a quanto accade con il Lego: per ottenere il braccio robot basta unire più pezzi assieme per formare elementi di maggior lunghezza. Altri componenti sono ruote dentate e viti senza fine per far girare gli ingranaggi. C'è anche una piccola base (mm. 260 x 187), sempre in plastica, che serve come base di appoggio per i vari apparecchi o, nel caso del plotter, come supporto per i fogli di carta. Il kit comprende anche elementi di controllo per la parte elettrica: 8 interruttori, svariati spinotti e prese, una coppia di motori elettrici, un cavo piatto a 20 vie, spezzoni di filo elettrico per i collegamenti e via dicendo.

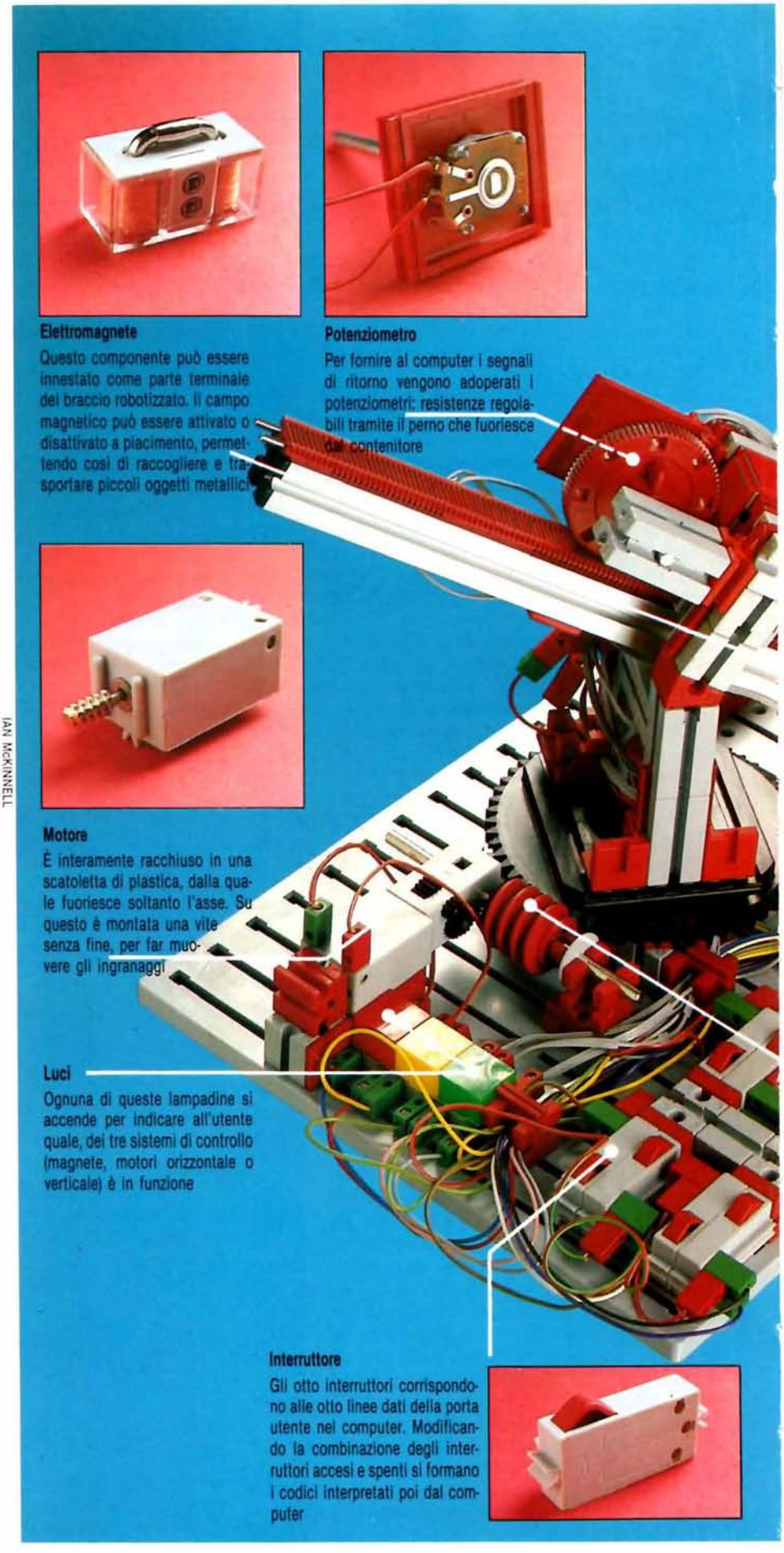
Tutti i componenti della scatola di montaggio sembrano sufficientemente robusti e compatti, quindi sembra garantita una lunga durata dei pezzi, anche dopo un uso prolungato. C'è una sola eccezione: la base di protezione del potenziometro (vedere figura) è relativamente fragile, se paragonata agli altri componenti. La rottura della base plastica non compromette direttamente il funzionamento del potenziometro, ma a lungo andare può causare problemi.

Un altro inconveniente del potenziometro è che il fissaggio dei cavetti ai suoi contatti non avviene, come per gli altri pezzi, a incastro: i fili, previamente sbucciati, vanno attorcigliati sui piedini del potenziometro e ciò può provocare, se non si fa attenzione, un falso contatto o anche un corto circuito. È consigliabile, per chi abbia dimestichezza con il saldatore, stagnare tre cavetti ai reofori del potenziometro, in modo da ottenere un collegamento stabile e duraturo.

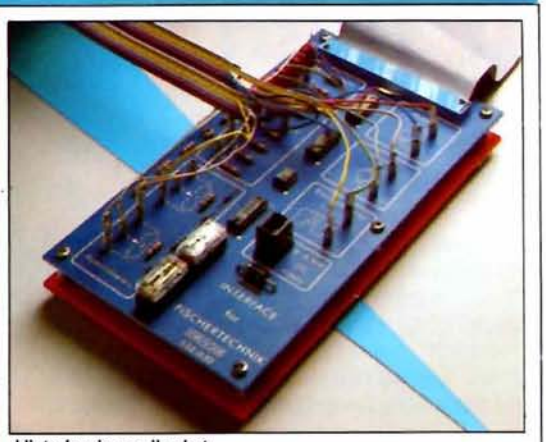
Montaggio dei componenti

La costruzione dei vari robot è piuttosto semplice e i pezzi sono maneggevoli anche per le dita meno abili. Gli inconvenienti maggiori non provengono dai pezzi del kit, quanto dal manuale, che è il punto più debole dell'intero sistema. La descrizione dei collegamenti è sufficientemente chiara, al contrario delle illustrazioni che mostrano le varie fasi di costruzione.

Sono riprodotte foto dei pezzi da utilizzare per ogni particolare progetto, assieme ad un "esploso" della realizzazione, ma il guaio è che mancano molti dettagli e scritte esplicative, lasciando il dubbio sull'esatta collocazione di alcuni pezzi.

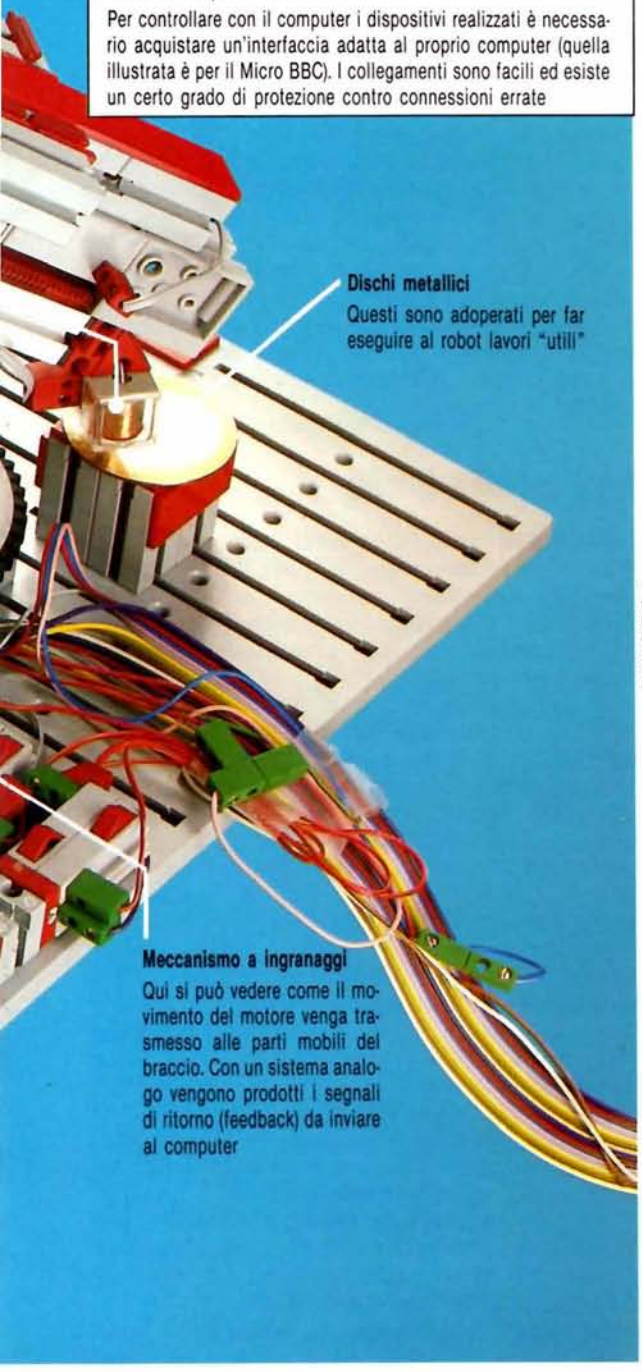


IAN McKINNELL



L'interfaccia per il robot

Per controllare con il computer i dispositivi realizzati è necessario acquistare un'interfaccia adatta al proprio computer (quella illustrata è per il Micro BBC). I collegamenti sono facili ed esiste un certo grado di protezione contro connessioni errate



Dischi metallici

Questi sono adoperati per far eseguire al robot lavori "utili"

Meccanismo a ingranaggi

Qui si può vedere come il movimento del motore venga trasmesso alle parti mobili del braccio. Con un sistema analogo vengono prodotti i segnali di ritorno (feedback) da inviare al computer

CHRIS STEVENS

Ad aggravare le cose, le foto sono in bianco e nero (ciò non aiuta affatto a stabilire il colore dei cavetti per i collegamenti) e spesso ritraggono il progetto da una sola angolazione, lasciando alla fantasia il compito di intuire il montaggio delle parti "in ombra". In conclusione, non sarebbe male se la Fischertechnik dedicasse un po' più di attenzione alla redazione del manuale, considerata l'elevata qualità di tutto il resto.

Infine, per collegare il robot al computer, occorre un'interfaccia adatta (venduta a parte), a seconda del computer usato. La connessione al computer, comunque, è molto semplice e, ad esempio sul BBC, avviene tramite tre porte separate: la porta per utente, quella analogica e quella della stampante. Per evitare eventuali danni alle apparecchiature (sia del kit che del computer), l'interfaccia è progettata in modo da impedire collegamenti elettrici pericolosi.

Il manuale dell'interfaccia (prodotta per la Fischertechnik dalla Unilab), ha una qualità nettamente superiore a quella del manuale per il kit e contiene utilissime informazioni sul controllo, da parte del computer, degli apparecchi realizzati.

Il linguaggio Prof

Assieme all'interfaccia viene fornita una cassetta che contiene tre programmi dimostrativi sul controllo dei dispositivi. I primi due riguardano il braccio robot e il plotter, mentre il terzo è il *Prof*, un sistema operativo specificamente progettato, che contiene numerosi comandi per il controllo delle apparecchiature realizzate con il kit. Essenzialmente si tratta di un'estensione del normale BASIC e molte delle parole riservate appartenenti al *Prof* sono nomi di variabili quali Motor, Magnet e altre, utili per semplificare il controllo.

Benché il *Prof* sia notevolmente completo, sotto il punto di vista della programmazione (anche strutturata), la maggior parte delle operazioni concerne l'emissione dei segnali di controllo. Rimane esclusa, pertanto, la gestione della tavoletta grafica realizzabile con il kit e, inoltre, il *Prof* non prevede la memorizzazione su nastro dei dati in ingresso. Ciò costringe a realizzare i propri programmi di controllo, ma non è un fatto del tutto negativo poiché parte del divertimento sta anche nella programmazione del computer.

Se si eccettuano gli inconvenienti citati causati dalla documentazione, il Robotics Kit è nel suo insieme un sistema di alto pregio didattico e applicativo. I robot, che si possono realizzare nel giro di poche ore, funzionano in modo eccellente. Inoltre è possibile costruire altri modelli, oltre ai sei suggeriti dal manuale.

Ad un prezzo forse troppo elevato per essere un giocattolo, questo sistema è certamente adatto a chi desidera accostarsi per la prima volta alla robotica.

FISCHER-TECHNIK ROBOTICS KIT

MOVIMENTO

I movimenti sono assicurati da due motori elettrici

CONTROLLO

Il controllo dei dispositivi realizzati può avvenire sia tramite il collegamento ad un computer, sia per mezzo degli otto interruttori che fanno parte del progetto

ALIMENTAZIONE

I motori vengono alimentati da una corrente continua a 6 volt. In genere vengono adoperate pile del tipo PJ996/4R25

INTERFACCE

Le interfacce necessarie per il controllo tramite computer (prodotte dalla Unilab) vengono vendute separatamente e variano a seconda del computer impiegato

DOCUMENTAZIONE

Il manuale del Kit è il punto debole del sistema: spesso le illustrazioni (in bianco e nero) non sono sufficientemente chiare

PRO

La scatola di montaggio offre la possibilità di realizzare più progetti dei sei suggeriti dal manuale, quindi il campo di sperimentazione è notevolmente ampio. I pezzi sono robusti e di facile innesto tra loro

CONTRO

Il prezzo di vendita è relativamente alto (del resto il kit non può essere considerato soltanto un giocattolo) e, in certi punti, il manuale è poco esauriente

In un mare di guai

Nel viaggio verso il Nuovo Mondo non mancano certo gli imprevisti

In questa fase vengono aggiunti i 6 eventi considerati "principali", in quanto i loro effetti sul viaggio sono molto più complessi e significativi di quelli introdotti fin qui. Come al solito, ogni evento può verificarsi una sola volta durante la traversata. All'inizio di ogni settimana, il programma seleziona a caso uno dei sei eventi, ma se questo si è già verificato durante il viaggio, la settimana trascorre senza alcun evento principale.
La linea 49 dimensiona la matrice M(), usata per indicare gli eventi già accaduti. L'istruzione GOSUB, alla linea 870 del ciclo principale, richiama la subroutine che seleziona un evento (linea 6500). La linea 6508 genera un numero casuale compreso tra 1 e 10: poiché gli eventi principali sono solo 6, ciascuno ha il 60% di probabilità di essere selezionato. Se vengono generati i numeri 7, 8, 9 o 10 non si verifica alcun evento. Tuttavia è possibile modificare le probabilità cambiando la gamma di numeri casuali generabili.
La linea 6508 può essere così modificata:

```
X=INT(RND(1)*8)+1
```

Adesso le probabilità che venga selezionato un evento principale passano al 75%. Pertanto, se viene selezionato un evento principale e il relativo marcatore nella matrice M() viene posto a 1, le probabilità che in seguito si verifichi un secondo evento calano al 62%, quelle di un terzo al 50% e così via. Quindi più eventi capitano, minore è la probabilità che se ne verifichino altri.
La subroutine dell'evento viene selezionata per mezzo di un'istruzione ON X GOSUB, simile a quella impiegata per gli eventi minori. Se X, ossia il numero generato in modo casuale, è 1, il programma salta al primo numero di linea nella lista di numeri di linea che segue l'istruzione, cioè alla linea 6530 (avvistamento di una scialuppa di salvataggio); se X è 2 si salta alla subroutine in 6700 (epidemia mortale). Per adesso ci limiteremo alla descrizione di questi due eventi: gli indirizzi delle altre 4 subroutine vengono aggiunti alla linea 6510 in un secondo momento.
Nella prima subroutine, il giocatore avvista in lontananza una scialuppa di salvataggio con dentro 4 uomini e una grossa cassa: deve decidere se prendere a bordo i naufraghi. Le operazioni di recupero allungano il viaggio di due giorni e portano a bordo altri quattro uomini da sfamare, ma questi possono però tornare comodi se il rendimento dell'equipaggio è basso. Per prima cosa il programma controlla se la scialuppa è stata avvistata in precedenza, esaminando (linea 6535) il primo elemento della matrice M(). Se questo vale 1, l'evento si è già verificato e il controllo torna al programma principale. Se il valore è 0, la linea 6540 pone M(1) a 1 per evitare che l'evento si verifichi nuovamente.
Quindi il programma chiede al giocatore se vuol prendere i naufraghi a bordo (linea 6576) e controlla se il primo carattere della risposta è S o N (linea 6580); in caso contrario torna a ripetere la domanda. Se la risposta è negativa (N), la scialuppa scompare all'orizzonte e il controllo torna al programma principale. Se il giocatore decide di salvare i naufraghi, il programma verifica se a bordo c'è spazio sufficiente, ossia se l'equipaggio è al completo. In caso affermativo (CN=16) il giocatore non può accogliere i naufraghi e il controllo torna al programma principale. La linea 6630 calcola gli "effettivi" mancanti assegnando a X il valore 16 meno il numero CN dei membri dell'equipaggio. Se X è maggiore di 3, sulla nave c'è spazio sufficiente per tutti i naufraghi; se X è 3 o minore di 3, possono essere accolti solo X uomini.

Naufraghi a bordo

I naufraghi vengono quindi registrati nella matrice di rendimento/tipo, TS(,), dell'equipaggio (linee 6638-6697). La variabile X, inizializzata a 0, viene usata per tenere il conto dei nuovi marinai. Quindi il programma verifica se gli elementi della matrice TS(,) contengono il tipo d'equipaggio 0, che indica uno spazio per un nuovo membro dell'equipaggio; in caso positivo, il valore di X viene incrementato di 1. Quando X raggiunge 4, tutti i naufraghi sono stati inseriti nella matrice dell'equipaggio e la linea 6655 assegna a T il valore 16, provocando l'uscita del programma dal ciclo alla successiva istruzione NEXT. Se c'è spazio per meno di quattro marinai, il programma abbandona il ciclo quando T raggiunge il valore 16. Il numero dei nuovi marinai viene sommato anche a quello dei membri dell'equipaggio, nella variabile CN. All'interno della matrice TS(,), ad ogni nuovo marinaio vengono assegnati due valori casuali: uno, compreso tra 1 e 5, relativo al tipo d'equipaggio; l'altro, tra 50 e 99, relativo al rendimento. Ciò significa che i nuovi uomini vengono considerati in buona o in ottima salute.
Poiché l'assegnazione del tipo d'equipaggio ai quattro naufraghi avviene in modo casuale, tra questi potrebbe trovarsi un dottore, non previsto nel reclutamento iniziale dell'equipaggio. La scialuppa di salvataggio contiene anche una cassa di provviste. La linea

IAN McKINNELL

6685 inizia un ciclo per i quattro tipi di provviste mentre la linea 6690 genera un numero casuale compreso tra 10 e 19 che ne definisce la quantità. La linea 6692 visualizza la quantità, le unità (ossia chili o barili registrati nella matrice U$()) e il tipo delle diverse provviste. Se una provvista si è esaurita durante la settimana la linea 6693 ne riporta la quantità da −999 a 0 consentendo una nuova registrazione; la linea 6694 somma le nuove provviste alle scorte esistenti.

Epidemia a bordo

Il secondo evento principale, la comparsa di un'epidemia, viene richiamato tramite il secondo numero di linea nell'istruzione ON X GOSUB, ossia 6700. Gli sviluppi dell'epidemia dipendono da due fattori: la presenza o meno a bordo di un medico e di medicine acquistate nella fase di approvvigionamento. Come in precedenza, il programma verifica che la subroutine non sia già stata usata e restituisce il controllo se l'elemento M(2) vale 1.
Quindi il programma controlla se c'è un medico a bordo. Un ciclo a 16 passaggi esamina la matrice del rendimento/tipo dell'equipaggio alla ricerca di un 2, corrispondente al medico. Se il dottore è ancora in vita (il suo rendimento non è 0 o −999), la variabile X viene azzerata. Se invece, alla fine del ciclo, X contiene il valore 1, non ci sono medici a bordo. Infine il programma controlla se ci sono scorte di medicinali, esaminando il valore di OA(1). In caso affermativo, Y viene posto a 0. Se, al termine della verifica, Y contiene il valore 1 non ci sono scorte di farmaci.
La linea 6730 crea un parametro "di malattia", Z, e usa la formula Z=((X+Y)*10)+5) per calcolare il calo di rendimento dell'equipaggio. I valori di X e Y dipendono dalla presenza di un medico e di medicinali: se entrambi sono a bordo il rendimento si riduce di 5. La mancanza del medico o di medicinali riduce il rendimento di 15; se entrambi sono assenti cala di 25. Le linee 6734 e 6736 comunicano gli effetti dell'epidemia, usando i valori di X e Y per verificare la presenza del medico o delle medicine.
Se alcuni uomini sono già molto debilitati, la malattia può ucciderli. La subroutine compresa tra le linee 6755 e 6775 esamina la matrice rendimento/tipo dell'equipaggio e riduce il rendimento dei marinai che hanno contratto la malattia. Tuttavia l'epidemia non colpisce tutto l'equipaggio. Infatti, se il numero casuale generato dalla linea 6756 è minore di 0.3, il rendimento del membro dell'equipaggio registrato nel corrente elemento della matrice non viene modificato. Quindi ogni marinaio ha il 30% di probabilità di non ammalarsi (questo fattore può essere modificato dal giocatore o generato in modo casuale). La linea successiva controlla se il marinaio è ancora in vita: in tal caso ne riduce il rendimento di Z unità. Se il rendimento cala al di sotto dello 0 il membro dell'equipaggio muore: il suo rendimento passa a −999 e il decesso compare nel rapporto di fine settimana. Infine il valore di X, ossia il numero degli uomini uccisi dall'epidemia, viene aumentato di 1. Se a bordo c'erano dei medicinali (Y=0), la linea 6776 ne dimezza la quantità arrotondandone il numero all'unità superiore. Se l'epidemia non uccide alcun membro dell'equipaggio (X=0), il controllo torna al programma principale. Altrimenti un messaggio comunica il numero di marinai deceduti.
Nella prossima parte vengono esaminati gli altri eventi principali e un problema relativo al timone della nave.



NONDUM

MODULO 8: DUE EVENTI PRINCIPALI

Dimensionamento della matrice per gli "eventi accaduti"

```
49 DIM M(6): REM SEGNALA GLI EVENTI GIA'
   ACCADUTI
```

Da inserire nel ciclo principale del viaggio

```
870 GOSUB 6500: REM ROUTINE DEGLI EVENTI
    PRINCIPALI
```

Routine di selezione degli eventi principali

```
6500 REM EVENTI PRINCIPALI
6508 X=INT(RND(1)*10)+1
6510 ON X GOSUB 6530,6700
6520 RETURN
```

Routine degli eventi principali

```
6530 REM SCIALUPPA DI SALVATAGGIO
6535 IF M(1)=1 THEN RETURN
6536 PRINT CHR$(147)
6540 M(1)=1
6550 S$="ALL'ORIZZONTE COMPARE UNA*":
     GOSUB 9100
6552 S$="SCIALUPPA DI SALVATAGGIO*":
     GOSUB 9100
6554 PRINT: GOSUB 9200
6556 S$="VEDI COL CANNOCCHIALE*": GOSUB 9100
6558 S$="CHE A BORDO CI SONO*": GOSUB 9100
6560 PRINT: GOSUB 9200
6562 S$="QUATTRO NAUFRAGHI*": GOSUB 9100
6563 GOSUB 9200
6564 S$="E UNA GROSSA CASSA!*": GOSUB 9100
6566 PRINT: GOSUB 9200
6568 S$="SE CAMBI ROTTA PER*": GOSUB 9100
6570 S$="SOCCORRERLI, IL VIAGGIO SI*":
     GOSUB 9100
6572 S$=" ALLUNGA DI DUE GIORNI!*":
     GOSUB 9100
6574 PRINT: GOSUB 9200
6576 S$="VUOI SALVARLI (S/N)?*": GOSUB 9100
6578 INPUT I$: I$=LEFT$(I$,1)
6580 IF I$<>"S" AND I$<>"N" THEN 6578
6585 IF I$="S" THEN 6600
6588 PRINT: GOSUB 9200
6590 S$="LA SCIALUPPA SCOMPARE...*":
     GOSUB 9100
6592 PRINT: GOSUB 9200
6594 S$=K$: GOSUB 91000
6596 GET I$: IF I$="" THEN 6596
6599 RETURN
6600 PRINT: GOSUB 9200
6610 EW=EW+2/7
6625 IF CN<>16 THEN 6630
6627 S$="NON PUOI PRENDERLI A*": GOSUB 9100
6628 S$="BORDO: NON C'E' SPAZIO.*":
     GOSUB 9100
6629 GOTO 6592
6630 X=16-CN: IF X>3 THEN6635
6632 S$="C'E' POSTO SOLO PER*": GOSUB 9100
6633 PRINT X;" UOMINI"
6634 PRINT: GOSUB 9200
6635 S$="NAUFRAGHI SALVATI:*": GOSUB 9100
6638 X=0
6640 FOR T=1 TO 16
6645 IF TS(T,1)<>0 THEN 6679
6650 X=X+1
6655 IF X>4 THEN T=16: GOTO 6679
6660 CN=CN+1
6665 TS(T,1)=INT(RND(1)*5)+1
6668 TS(T,2)=INT(RND(1)*50)+50
6670 PRINT"1 ";C$(TS(T,1))
6679 NEXT
6680 PRINT: GOSUB 9200
6682 S$="LA CASSA CONTIENE:*": GOSUB 9100
6685 FOR T=1 TO 4
6690 X=INT(RND(1)*10)+10
6692 PRINT X;U$(T);" DI ";P$(T)
6693 IF PA(T)=-999 THEN PA(T)=0
6694 PA(T)=PA(T)+X
6695 NEXT
6699 GOTO 6592
```

```
6700 REM EPIDEMIA MORTALE
6705 IF M(2)=1 THEN RETURN
6706 PRINT CHR$(147)
6710 M(2)=1
6712 S$="EPIDEMIA TRA L'EQUIPAGGIO!*":
     GOSUB 9100
6714 PRINT: GOSUB 9200
6716 X=1
6718 FOR T=1 TO 16
6720 IF TS(T,1)=2 AND TS(T,2)<>0 AND TS
     (T,2)<>-999 THEN X=0: T=16
6722 NEXT
6724 Y=1
6726 IF OA(1)<>0 AND OA(1)<>-999 THEN Y=0
6730 Z=((X+Y)*10)+5
6732 I$="A BORDO CI SONO ": IF X=0 AND Y=0
     THEN 6740
6734 IF X=1 THEN S$="MA NESSUN MEDICO*":
     GOSUB 9100: I$=" E "
6736 IF Y=1 THEN S$= I$+"NESSUNA MEDICINA*":
     GOSUB 9100
6740 S$="MOLTI UOMINI SI AMMALANO"*:
     GOSUB 9100
6745 PRINT: GOSUB 9200
6750 X=0
6755 FOR T=1 TO 16
6756 IF RND(1)<.3 THEN 6775
6760 IF TS(T,2)=0 OR TS(T,2)=-999 THEN 6775
6765 TS(T,2)=TS(T,2)-Z
6770 IF TS(T,2)<1 THEN TS(T,2)=-999: X=X+1
6775 NEXT
6776 IF Y=1 THEN 6780
6777 S$="META' DELLE MEDICINE ESAURITA*":
     GOSUB 9100
6778 OA(1)=INT((OA(1)/2)+.5)
6780 PRINT: GOSUB 9200
6785 IF X=0 THEN 6797
6790 PRINT "E";X;
6792 S$="MEMBRI MORTI*"
6794 IF X=1 THEN S$="MEMBRO MORTO*"
6795 GOSUB 9100
6796 PRINT: GOSUB 9200
6797 S$=K$: GOSUB 9100
6798 GET I$: IF I$="" THEN 6798
6799 RETURN
```

Basic & Basic

Spectrum:

Cambiare le seguenti linee:

```
6510 IF X=1 THEN GO SUB 6530
6511 IF X=2 THEN GO SUB 6700
6536 CLS
6578 INPUT I$: LET I$=I$(1 TO 1)
6596 LET I$=INKEY$: IF I$="" THEN GO TO 6596
6706 CLS
6798 LET I$=INKEY$: IF I$="" THEN GO TO 6798
```

Micro BBC:

Cambiare le seguenti linee:

```
6536 CLS
6596 I$=GET$
6706 CLS
6798 I$=GET$
```

Olio di gomito

Il programma di controllo del braccio robot, per Commodore 64

Il sistema operativo del Commodore genera una serie regolare di interrupt, che governano le operazioni di sistema (quali la scansione della tastiera). Tali interrupt si verificano a intervalli di un sessantesimo di secondo. La sezione in codice macchina del programma per il Commodore intercetta il vettore dell'interrupt ed esegue una propria routine, prima di restituire il controllo alla normale routine dell'interrupt. Questo programma prende il nome di" cuneo" (vedere a pag. 1619) e la sua prima parte sostituisce il normale vettore IRQ con l'indirizzo della routine occorrente. Effettuata la sostituzione, la nostra routine di gestione degli eventi viene eseguita ogni volta che si verifica un'interrupt IRQ. Il programma di gestione degli eventi, assai simile a quello di controllo multiplo dei servomotori presentato a pag. 1403, trasmette ai motori gli impulsi di corrente provenienti dalla porta utente attraverso le linee per i dati. La durata di ogni impulso determina l'angolo di rotazione del motore corrispondente. La routine in codice macchina riceve l'informazione relativa alla durata degli impulsi per i motori da una tabella ad 8 byte denominata ANGOLO. Come accade nella versione per il Micro BBC, una sezione di programma supplementare usa una seconda tabella, chiamata NEWPOS, per caricare le locazioni di ANGOLO. NEWPOS viene a sua volta caricata dal programma BASIC di controllo.

Il programma BASIC che controlla i movimenti del braccio ricorre alla sezione in codice macchina appena descritta per pilotare il braccio robot. Come nella versione per il Micro BBC, il programma permette di controllare il bracco agendo sulla tastiera e di memorizzare le posizioni chiave, che possono essere poi ripetute a piacimento. Tra la versione Commodore e quella Micro BBC esistono due sostanziali differenze. Innanzi tutto, durante la fase di controllo del braccio, i motori tendono ad oscillare rispetto alla posizione programmata.

Ciò avviene perché il chip video VIC-II del Commodore interferisce con la regolare generazione delle interrupt.

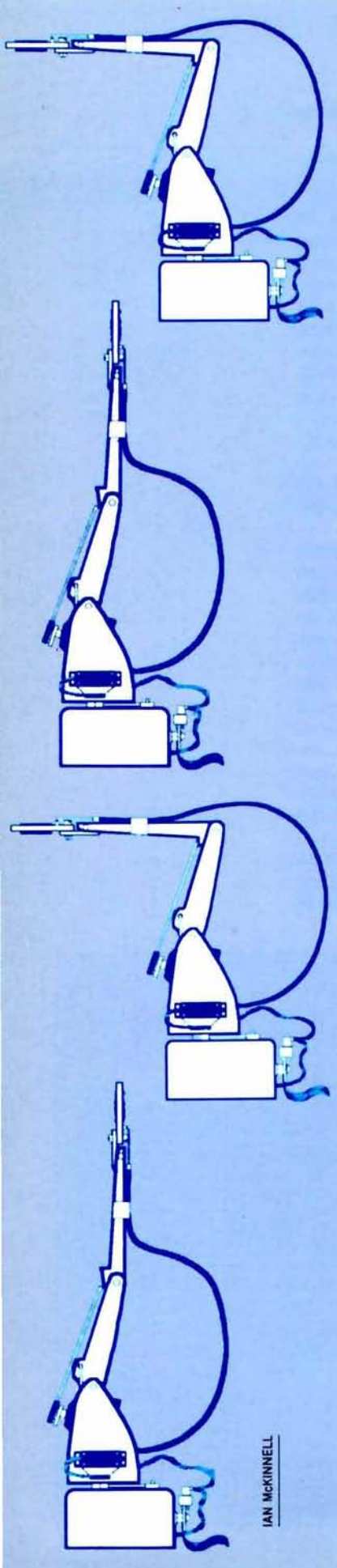
Il modo migliore per eliminare questo difetto è quello di disabilitare lo schermo durante l'esecuzione della routine cuneo. Le subroutine alle linee 7000 e 8000 eseguono tale operazione prima dell'inserimento del cuneo e ripristinano l'immagine al termine della routine.

Un secondo problema è che il BASIC del Commodore esegue la verifica dei numerosi tasti impiegati (e le conseguenti operazioni di salto) con eccessiva lentezza. L'ultima subroutine confronta il codice ASCII di ogni tasto battuto con quelli contenuti nella tabella dei tasti che controllano i movimenti del braccio. Quando il risultato è positivo il codice ASCII in esame viene passato come un numero che può essere usato da un'istruzione ON... GOSUB del BASIC per richiamare la giusta routine.

PROGRAMMA DI CONTROLLO DEL BRACCIO PER IL COMMODORE 64

Caricatore del BASIC

```
 10 REM ****************************
 20 REM **                        **
 30 REM ** CARICATORE BASIC PER   **
 40 REM ** CONTROLLO DEL BRACCIO  **
 50 REM **      VERSIONE CBM      **
 55 REM **                        **
 60 REM ****************************
 65 :
 70 FOR I=49155 TO 49394
 75 READ A: POKE I,A:CC=CC+A
 80 NEXT I
 90 READ CS:IF CS<>CC THEN PRINT
    "ERRORE DI CHECKSUM":STOP
 95 :
100 DATA 169,255,141,3,221,169,62,141,1
110 DATA 192,169,192,141,2,192,120,173
120 DATA 20,3,174,1,192,141,1,192,142
130 DATA 20,3,173,21,3,174,2,192,141,2
140 DATA 192,142,21,3,169,16,133,251
150 DATA 169,193,133,252,169,255,160,0
160 DATA 145,251,136,208,251,88,96,8,72
170 DATA 152,72,138,72,169,255,141,1
180 DATA 221,162,7,169,255,24,106,72
190 DATA 188,0,193,49,251,145,251,104
200 DATA 202,16,243,160,48,136,208,253
210 DATA 169,255,160,0,49,251,141,1,221
220 DATA 200,208,248,162,7,169,255,188
230 DATA 0,193,145,251,202,16,248,104
240 DATA 170,104,168,104,40,108,1,192
250 DATA 162,0,160,0,189,8,193,221,0
260 DATA 193,240,14,176,6,222,0,193,76
270 DATA 156,192,254,0,193,76,156,192
280 DATA 200,232,224,4,208,228,32,169
290 DATA 192,192,4,208,217,96,138,72
300 DATA 152,72,160,255,174,0,192,136
310 DATA 234,234,208,251,202,208,248
320 DATA 104,168,104,170,96,0,255,255,0
330 DATA 0,255,255,0,0,255,255,0,0,255
340 DATA 255,0,0,72,138,72,152,72,32
350 DATA 228,255,201,0,240,249,162,0
360 DATA 221,191,192,240,7,232,224,17
370 DATA 208,246,162,255,142,207,192
380 DATA 104,168,104,170,104,96
390 DATA 33097:REM *CHECKSUM*
```



IAN McKINNELL

Listato Assembly

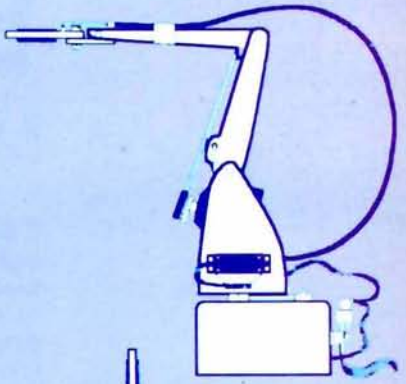
```
;************************
;************************
;++   CONTROLLO DEL    ++
;++     BRACCIO        ++
;++   VERSIONE CBM     ++
;************************
;************************
;
PORTA  = 56577     ;REGISTRO DATI PORTA UTENTE
DDR    = 56579     ;REGISTRO DIREZ. DATI PORTA UTENTE
ANGOLO = $C100     ;LOCAZIONE PER VALORE ANGOLO
MEMPOS = $C108     ;TABELLA POSIZIONI
TABMOT = $C110     ;TABELLA PER IL MOTORE
PAGZ   = $FB       ;PUNTATORE PAGINA 0 ALLA TABELLA
IRQVEC = $0314     ;VETTORE IRQ (PARTE BASSA)
;
*=$C000
FATRIT *=*+1       ;DEPOSITO PER FATTORE RITARDO
NUOVET *=*+2       ;DEPOSITO PER IL VETTORE
       LDA #$FF
       STA DDR     ;DDR SELEZIONATO IN USCITA
       LDA #<EVENTO
       STA NUOVET  ;PUNTATORE ALLA ROUTINE
       LDA #>EVENTO ;PER TRATTAMENTO EVENTI
       STA NUOVET+1
;
       SEI         ;INTERRUPT DISABILITATE
       LDA IRQVEC  ;SCAMBIA VETTORE IRQ
       LDX NUOVET  ;ORIGINALE CON
       STA NUOVET  ;QUELLO NUOVO
       STX IRQVEC
       LDA IRQVEC+1
       LDX NUOVET+1
       STA NUOVET+1
       STX IRQVEC+1
;
;+++++ INIZIALIZZA LA TABELLA +++++
;
       LDA #<TABMOT
       STA PAGZ
       LDA #>TABMOT
       STA PAGZ+1
;
       LDA #$FF
       LDY #$00
TABEL
       STA (PAGZ),Y
       DEY
            BNE TABEL
       CLI         ;ABILITAZIONE INTERRUPT
       RTS
;
;+++++ TRATTAMENTO EVENTI +++++
;
EVENTO
       PHP
       PHA         ;SALVA I REGISTRI
       TYA         ;SULLO STACK
       PHA
       TXA
       PHA
;
;+++++ CONTROLLO TASTIERA +++++
;
TABTAS *=*+16
RISULT *=*+1
GETIN=$FFE4
;
VERIF
       PHA
       TXA
       PHA
       TYA
       PHA
NOTAST
       JSR GETIN   ;ESTRAE CARATTERE
       CMP #$00    ;NESSUN CARATTERE?
       BEQ NOTAST
       LDX #$00
CONFR
       CMP TABTAS,X
       BEQ USCITA  ;CARATTERE IN TABELLA?
       INX
       CPX #17
       BNE CONFR
       LDX #$FF    ;SEGNALE NON TROVATO
USCITA
       STX RISULT  ;CONSERVA OFFSET NELLA
       PLA         ;TABELLA DEI TASTI
       TAY
       PLA
       TAX
       PLA
       RTS
       BNE CICLO
;
       LDX #$07
       LDA #$FF
CLEAR
       LDY ANGOLO,X   ;RIMUOVE TUTTE LE ECCEZIONI
       STA (PAGZ),Y
       DEX         BPL CLEAR
;+++++ IMPULSI TERMINATI +++++
;
       PLA
       TAX
       PLA         ;RIPRISTINA REGISTRI
       TAY
       PLA
       PLP
       JMP (NUOVET)   ;SALTA ALLA NORMALE
                      ;ROUTINE IRQ
CICLO2
       LDX #$00
       LDY #$00
CICLO1
       LDA MEMPOS,X   ;NUOVA POSIZIONE PER SERVO X
       CMP ANGOLO,X   ;UGUALE A VECCHIA POSIZIONE?
       BEQ ROSSO      ;SI, NESSUNA AZIONE
       BCS SOMMA      ;SE ) ALLORA SOMMA
       DEC ANGOLO,X   ;ALTRIMENTI SOTTRAE
       JMP INCRX
SOMMA
       INC ANGOLO,X
       JMP INCRX
ROSSO
       INY
INCRX
       INX
       CPX #$04       ;OPERATO SU TUTTI I 4 SERVO?
       BNE CICLO1     ;NO, PASSA AL PROSSIMO
       JSR PAUSA1     ;BREVE PAUSA
       CPY #$04       ;SE Y=4 ALLORA TUTTO FATTO
       BNE CICLO2
       RTS
;
PAUSA1
       TXA
       PHA            ;SALVA REGISTRI  X & Y
       TYA
       PHA
       LDY #$FF
       LDX FATRIT     ;ESTRAE FATTORE DI RITARDO
ANCORA
       DEY
       NOP            ;256*8 CICLI DI CLOCK
       NOP            ;ATTENDE
       BNE ANCORA
       DEX            ;RIPETE SE E' PAUSA
       BNE ANCORA     ;ANCORA NECESSARIA
       PLA
       TAY
       PLA
       TAX
       PHA
;
;++ IMPULSO D'INIZIO. CON ALCUNI MOTORI E'
;POSSIBILE INIZIARE PRIMA DI RIEMPIRE LA
;TABELLA: DIMINUIRE IL CICLO DI ATTESA QUI
;SOTTO ++
;
       LDA #$FF
       STA PORTA
;++ RIEMPIE TABELLA CON LE ECCEZIONI ++
       LDX #$07
       LDA #$FF
       CLC
ECCEZ
       ROR A
       PHA            ;CONFIGURAZIONE DI BIT
       LDY ANGOLO,X   ;ESTRAE OFFSET X PER IL MOTORE
       AND (PAGZ),Y   ;SALVA CONFIGURAZIONE ESISTENTE
       STA (PAGZ),Y   ;MA MODIFICA QUELLA X DEL MOTORE
       PLA
       DEX
       BPL ECCEZ
;+++++ TABELLA RIEMPITA +++++
       LDY #$30
PAUS
       DEY            ;CICLO DI RITARDO
       BNE PAUS       ;PER UNA PAUSA

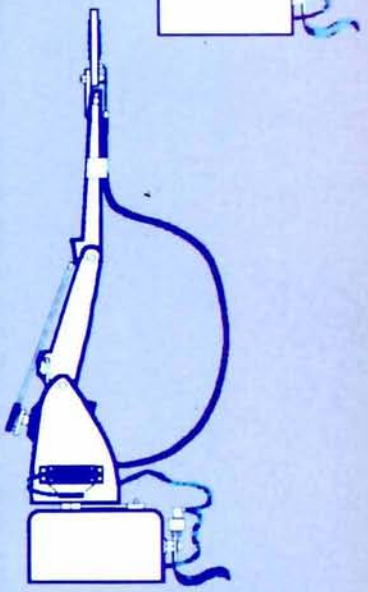
       LDA #$FF       ;TUTTI GLI IMPULSI ATTIVI
       LDY #$00
CICLO
       AND (PAGZ),Y   ;VENGONO OPPORTUNAMENTE
       STA PORTA      ;MASCHERATI CON CIASCUN
       INY            ;ELEMENTO DELLA TABELLA
```

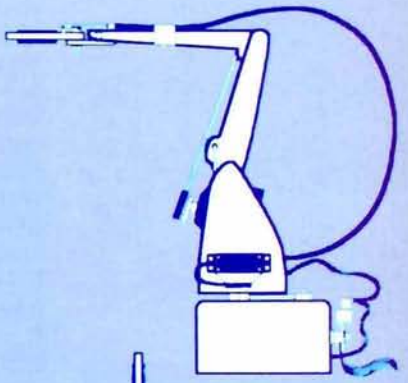
Programmatore di sequenze del braccio robot

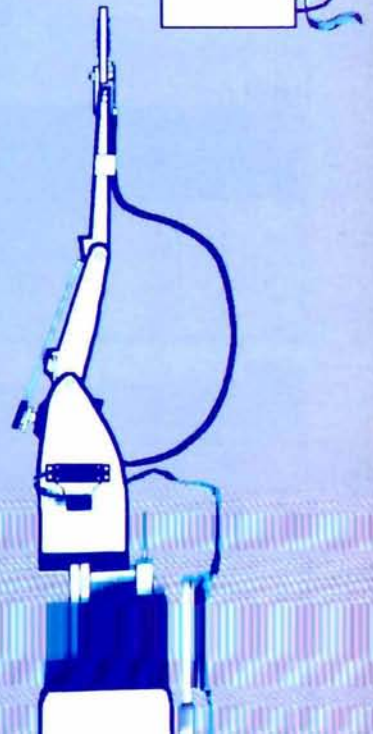
```
10 REM ********************************
20 REM ********************************
30 REM **                            **
40 REM **      PROGRAMMATORE DI      **
50 REM **  SEQUENZE PER IL BRACCIO   **
70 REM **        VERSIONE CBM        **
75 REM **                            **
80 REM ********************************
90 REM ********************************
95 :
96 DN=8: REM PER CASS DN=1
97 IF A=0 THEN A=1:LOAD"ROBARM.HEX",DN,1
100 GOSUB 1000:REM PREPARAZIONE
110 PRINT CL$
120 X=5:Y=6:GOSUB10000:PRINT"OPZIONI :"
130 X=4:Y=10:GOSUB10000:PRINT"1.....NUOVA
    SEQUENZA"
140 X=4:Y=12:GOSUB10000:PRINT"2.....AGGIUNTA
    MOVIMENTI"
150 X=4:Y=14:GOSUB10000:PRINT"3.....
    ESECUZIONE"
160 X=4:Y=16:GOSUB10000:PRINT"4.....FINE
    LAVORO"
170 GET G$:IF G$="" THEN 170
180 IF G$="1" THEN C=0:LM=0:REM PUNTATORI
    POSIZIONE DI RIPOSO
190 IF G$="1" OR G$="2" THEN GOSUB2000:
    GOSUB3000:GOSUB4000:GOSUB5000
200 IF G$="3" THEN GOSUB6000:GOSUB4000
210 IF G$="4" THEN PRINTCL$:END
220 GOTO 110
500 REM **** FUNZIONI ABBINATE AI TASTI ****
540 DX=DX+1:RETURN
550 DX=DX-1: IF DX<0 THEN DX=0
555 RETURN
560 P=PEEK(NP)+DX:IF P<256THEN POKENP,P
565 RETURN
570 P=PEEK(NP)-DX:IFP>0THEN POKENP,P
575 RETURN
580 P=PEEK(NP+1)+DX:IF P<256THEN POKENP+1,P
585 RETURN
590 P=PEEK(NP+1)-DX:IF P>0THEN POKENP+1,P
595 RETURN
600 P=PEEK(NP+2)-DX:IF P>0THEN POKENP+2,P
605 RETURN
610 P=PEEK(NP+2)+DX:IF P<256 THEN POKE
    NP+2,P
615 RETURN
620 P=PEEK(NP+3)+3*DX:IF P<256THEN POKE
    NP+3,P
625 RETURN
630 P=PEEK(NP+3)-3*DX:IF P<256THEN POKE
    NP+3,P
635 RETURN
640 FOR I=0 TO 3:POKE NP+1,R%(C,I):NEXT
    I:RETURN
650 C=C+1:IF C>MC THEN C=0
655 RETURN
660 C=C-1:IF C<0 THEN C=MC
665 RETURN
670 FOR I=0 TO 3:R%(C,I)=PEEK(NP+I):NEXT:
    C=C+1:RETURN
680 FOR I=0 TO 3:R%(C,I)=PEEK(NP+I):NEXT:
    C=C+1:RETURN
999 :
1000 REM ***** PREPARAZIONE *****
1010 S=0:NS=3:REM N. DI SERVOMOTORI-1
1020 LM=0:DC=0:DF=10:REM FATTORE DI RITARDO
1030 MC=100:REM CONTEGGIO MASSIMO
1040 DIM R%(MC,NS):REM MATRICE POSIZIONI DEI
    TASTI
1050 CL$=CHR$(147):REM PULIZIA SCHERMO
1060 DW$="":FOR I=1 TO 25:DW$=DW$+CHR$(17):
    NEXT I:REM CURSORE IN BASSO
1065 POKE 650,128:REM RIPETIZIONE AUTOMATICA
    DEI TASTI
1070 DL=49152:NP=49416:REM INDIRIZZI PER
    RITARDO E POSIZIONE
1075 GS=49155:OFF=49170:REM INDIRIZZI ROUTINE
    DI CUNEO
1077 MOVE=49281:REM INDIRIZZI ROUTINE DI
    MOVIMENTO
1078 TABTAS=RISULT=49359:REM INDIRIZZI
    PRESSIONE TASTI
1079 VERIF=REM INDIRIZZO VERIFICA TASTI
1080 REM ***** LEGGE ASCII DA TASTIERA *****
1090 FOR I=0 TO 14:READA:POKE TABTAS+1,A:
    NEXT I
1100 DATA 73,68,157,29,145,17,65,90,88,67
1110 DATA 82,78,66,83,81
1200 REM ***** AZZERA LOCAZIONI NEWPOS *****
1210 FOR I=0 TO 3:POKENP+I,0:NEXT I
1990 RETURN
1999 :
2000 REM ***** INFORMAZ. ****
2010 PRINTCL$
2020 X=5:Y=5:GOSUB10000:PRINT"USARE  :"
2060 X=1:Y=13:GOSUB10000:PRINT"S...PER
    MEMORIZZARE POSIZIONE"
2070 X=1:Y=15:GOSUB10000:PRINT"Q...PER
    TORNARE AL MENU"
2080 X=1:Y=17:GOSUB10000:PRINT"R...MOVIM.
    A POSIZ. MEMORIZZ."
2090 X=1:Y=19:GOSUB10000:PRINT"N & B...
    CONTATORE AVANTI/INDIETRO"
2100 X=1:Y=21:GOSUB10000:PRINT"E...NUOVO
    CONTATORE"
2110 X=1:Y=23:GOSUB10000:PRINT"I & D...INCR.
    /DECR. VELOCITA'"
2120 X=2:Y=2:GOSUB10000:PRINT"CONTO=";C;" ";
2130 FOR I=0 TO 3:PRINTR%(C,I);" ";:NEXT:PRINT
2135 GET A$:IF A$="" THEN 2135
2140 RETURN
2999 :
3000 REM ***** PROGRAMMA BRACCIO *****
3005 GOSUB7000:REM PULIZIA SCHERMO E INSERIM.
    CUNEO
3010 POKE DL,1:REM FATTORE DI RITARDO = 1
3020 DX=8:REM MODIFICA RITMO
3030 SYS VERIF:REM VERIFICA PRESSIONE TASTO
3035 R=PEEK(RISULT)+1:IF R>15 THEN 3030:TASTO
    NON VALIDO
3037 ON R GOSUB 540,550,560,570,580,590,600,
    610,620,630,640,650,660,670
3190 IF C>LM THEN LM=C-1:REM N. DI RECORD
    CORRENTE
3220 DC=C
3260 SYS MUOVI:REM ROUTINE DI MOVIMENTO IN C/M
3270 IF R=15 OR C>MC THEN GOSUB8000:RETURN:REM
    RIMUOVE CUNEO E FINE
3280 GOTO 3030:REM RIPETE
3999 :
4000 REM ***** RIPETIZIONE DI SEQUENZA *****
4010 PRINTCL$
4020 X=5:Y=23:GOSUB10000:PRINT"RIPETO SEQUENZA
    (S/N),R=RIPETO"
4030 GET AN$:IF AN$<>"S" AND AN$<>"N" AND AN$<>
    "R" THEN 4030
4040 IF AN$="N" THEN RETURN
4050 IF AN$<>"R" THEN X=5:Y=22:GOSUB10000:
    INPUT"FATTORE RITARDO 1-255";DF
4060 IF DF<0 OR DF>255 THEN 4050 :REM VERIFICA
    VALIDITA'
4070 POKE DL,DF:REM MODIFICA REGISTRO DEL
    RITARDO
4075 GOSUB7000:REM PULISCE SCHERMO E INSER.
    CUNEO
4080 FOR I=0 TO LM
4090 X=5:Y=2:GOSUB10000:PRINT"N. DEL PASSO =
    ";I;" "
4100 FOR S=0 TO 3
4110 POKE NP+S,R%(I,S)
4120 NEXT S:SYS MUOVI:REM ROUTINE MOVIM. IN C/M
4130 NEXT I
4135 GOSUB8000:REM RIPRISTINA SCHERMO E TOGLIE
    CUNEO
4140 GOTO 4010:REM RIPETE
4999 :
5000 REM ***** MEMORIZZA SEQUENZA *****
5010 PRINTCL$
5020 X=5:Y=10:GOSUB10000:PRINT"MEMORIZZATO SU
    FILE (S/N)"
5030 GET AN$:IF AN$<>"S" AND AN$<>"N" THEN 5030
5040 IF AN$="N" THEN RETURN
5050 X=5:Y=12:GOSUB10000:INPUT"NOME FILE";FL$
5060 OPEN 3,DN,3,"@0:"+FL$+".DATA,S,W":REM OPEN
    DEL FILE
5070 PRINT#3,LM
5080 FOR A=0 TO LM:FOR B=0 TO 3
5090 PRINT#3,R%(A,B):NEXT B,A
5100 PRINT#3:CLOSE3
5120 RETURN
5999 :
6000 REM ***** CARICA UN FILE *****
6010 PRINTCL$
6020 X=5:Y=10:GOSUB10000:INPUT"NOME FILE DA
    CARICARE";FL$
6040 OPEN 3,DN,3,"@0:"+FL$+".DATA,S,R":REM
    OPEN DEL FILE
6050 INPUT#3,C:LM=C
6060 FOR A=0 TO C:FOR B=0 TO 3
6070 INPUT#3,R%(A,B):NEXT B,A
6080 PRINT#3:CLOSE3
6100 RETURN
6999 :
7000 REM ***** INSERIM. CUNEO *****
7010 POKE53265,PEEK(53265)AND 239:REM PULISCE
    SCHERMO

7015 SYS GS
7020 RETURN
7999 :
8000 REM ***** RIMOZIONE CUNEO *****
8010 POKE53265,PEEK(53265)OR 16:REM PULISCE
    SCHERMO
```









Effetti ottici

Il funzionamento del chip del Commodore 64

Il Commodore 64 possiede 8 modi "base" per la grafica dello schermo. Nella grafica in bassa risoluzione, il "set" di caratteri può risiedere in ROM o in RAM ed essere visualizzato in uno dei tre modi seguenti: Standard, Multicolour e Extended colour; in alta risoluzione, i modi sono due: Standard e Multicolour. Oltre a questi modi base, il Commodore 64 consente variazioni di altro tipo: lo schermo può essere configurato su 38 colonne, anziché sulle regolari 40, e su 24 righe invece che sulle normali 25. Di solito tali variazioni vengono adoperate assieme alla funzione di scroll omogeneo orizzontale e verticale dello schermo. Sul Commodore 64, questa funzione produce risultati migliori se si ricorre al codice macchina, in quanto la progressiva sostituzione dell'immagine deve avvenire con grande rapidità all'interno della RAM del video. A pagina 1517 si è visto un programma di scorrimento orizzontale che usava per lo schermo il modo a 38 colonne.

Se viene usato lo schermo ad alta risoluzione in un programma di notevoli dimensioni, la memoria diventa un bene prezioso. Tuttavia il Commodore 64 consente una notevole libertà di movimento nell'indirizzamento degli schermi ad alta e bassa risoluzione. Come si vedrà in seguito, se il tracciamento viene eseguito in codice macchina, è possibile collocare uno schermo ad alta risoluzione nella RAM "nascosta" dalla ROM dell'interprete BASIC (vedere a pag. 1617). L'utilità di una simile soluzione è ovvia: se il programma è esclusivamente in codice macchina, l'interprete ROM occupa inutilmente ben 8 Kbyte di preziosa memoria! Per una coincidenza, 8 Kbyte è proprio l'area di memoria necessaria per uno schermo ad alta risoluzione.

Il Chip d'interfaccia per lo schermo 6566/67 (più brevemente: VIC-II) ha il compito di generare i dati per l'immagine video, che vengono poi inviati al televisore o al monitor. Per eseguire questa operazione, il VIC-II deve essere capace di leggere tali dati dalla RAM o dalla ROM. Vale la pena di analizzare come il VIC-II ottenga le proprie informazioni, ricordando che questo è uno dei processi meno evidenti del Commodore 64.

Tabella 1

Config. binaria della coppia di pixel	Colore	Determinato da
0 0	Sfondo dello schermo	53281 ($D021) Bit 0-4
0 1	Multicolour#1	53282 ($D022) Bit 0-4
1 0	Multicolour#2	53283 ($D023) Bit 0-4
1 1	Colore di primo piano	Bit 0-3 del "nybble" di colore

Tabella 2

Cod. di schermo	Bit 7	Bit 6	Registro colore di sfondo
64 — 127	0	1	53282 ($D022) Extended colour 1
128 — 191	1	0	53283 ($D023) Extended colour 2
192 — 255	1	1	53284 ($D024) Extended colour 3

Modo Multicolour

Lo schermo in bassa risoluzione e il modo "bit map" sono stati descritti in precedenza (vedere alle pagg. 694 e 817), pertanto qui vengono esaminati due degli altri modi previsti per la grafica su Commodore 64.

Il modo Multicolour consente di visualizzare 4 colori diversi, anziché 2 soli, in un'unica cella di carattere. L'unico inconveniente è che la risoluzione orizzontale diventa a coppie di pixel e non a pixel singoli. Il modo grafico Multicolour può essere usato sia in bassa sia in alta risoluzione, ma nel secondo caso la determinazione dei colori dei punti è un po' diversa. Queste istruzioni BASIC abilitano e disabilitano il modo Multicolour:

```
POKE 53270, PEEK(53270) OR 16
POKE 53270, PEEK(53270) AND 239
```

Se il modo Multicolour viene usato in bassa risoluzione e se il bit 4 del "nybble" (4 bit o semibyte) del colore associato vale 1, allora il carattere viene interpretato secondo il modo Multicolour e i tre bit più bassi ne definiscono il colore. Ciò significa che i caratteri con "nybble" di colore associato compresi tra 0 e 7 vengono visualizzati normalmente, mentre se il codice di colore è compreso tra 8 e 15, il carattere viene visualizzato nel modo Multicolour. I colori delle coppie di pixel vengono definiti secondo lo schema della Tabella A.

Agendo sul contenuto degli indirizzi 53282 e 53283 è possibile cambiare istantaneamente il colore di qualsiasi coppia di pixel Multicolour associati. È da notare che il modo Multicolour produce risultati migliori se usato su caratteri definiti dall'utente, poiché le configurazioni di bit traggono vantaggio dal fatto di essere interpretate come coppie di pixel.

Un altro modo grafico disponibile sul Commodore 64 è il modo Extended colour (a colori estesi), che permette di controllare il colore di sfondo nei primi 64 caratteri del set. Questo modo, che non può essere usato assieme al Multicolour, viene abilitato e disabilitato dalle seguenti linee BASIC:

```
POKE 53265, PEEK(53265) OR 64
POKE 53265, PEEK(53265) AND 191
```

Lo stesso carattere può apparire sullo schermo con un massimo di 4 diversi colori di sfon-

do, uno dei quali è il colore di sfondo dello schermo. I caratteri successivi al sessantaquattresimo non possono essere visualizzati nel modo Extended colour, perché i bit 6 e 7 del codice di schermo vengono usati per controllare indirettamente il colore del carattere. Il codice di schermo per la 'A', ad esempio, è 1 mentre quello di una 'A' in campo inverso è 65. Tuttavia un'istruzione POKE 65, se lo schermo è selezionato nel modo Extended colour, non produce una 'A' in campo inverso, bensì una 'A' normale con un colore di sfondo definito dal contenuto dell'indirizzo 53282 ($D022). Anche con l'istruzione POKE 129 si ottiene una 'A' normale ma con un colore di sfondo definito dal contenuto dell'indirizzo 53283 ($D023). La Tabella 2 mostra la corrispondenza tra i codici di schermo e i registri di colore.

Gestione della memoria

Il VIC-II "vede" in modo diverso e molto più semplice la mappa di memoria usata dal 6510. Di volta in volta, il VIC-II può accedere soltanto a uno dei 4 banchi di memoria da 16 Kbyte della memoria. Il VIC-II esamina quindi la memoria come attraverso una "finestra", le cui dimensioni sono un blocco di 16 Kbyte. L'indirizzo di base di questa finestra è controllato dal software e può assumere quattro valori diversi.

```
1000 REM ** SELEZIONA IL BANCO PER LA FINESTRA DEL VIC **
1010 POKE 56578, PEEK(56578) OR 3: REM DDR CIA£2 BIT 0,1 PER L'OUTPUT
1020 FIN = 3: REM SELEZIONA LA FINESTRA NORMALE
1030 POKE 56576, (PEEK(56576) AND 252) OR FIN: REM PORTA A CIA # 1 BIT 0,1
```

In questo caso FIN può assumere valori compresi tra 0 e 3. In qualsiasi momento l'istruzione PEEK(56576) AND 3 permette di conoscere il valore corrente di FIN. L'indirizzo in memoria, corrispondente all'inizio della finestra a 16 Kbyte, può essere calcolato con la formula VFIN = 16384*(3 — FIN).
Ai valori registrati in FIN corrispondono i seguenti indirizzi:

FIN	Indirizzo iniziale della finestra
0	49152 ($C000)
1	32768 ($8000)
2	16384 ($4000)
3	0 ($0000)

All'interno della finestra del VIC-II il 6510 si aspetta di trovare la memoria video e un'immagine formata da caratteri ROM in bassa risoluzione (o valori numerici per l'alta risoluzione se questa è stata selezionata), come le informazioni relative agli sprite. I puntatori a ciascuno degli otto sprite si trovano alla fine della memoria video (devono quindi essere spostati assieme allo schermo).
Lo scarto (offset) dell'inizio della memoria video rispetto alla base della finestra corrente del VIC-II è controllato dai quattro bit superiori del registro di controllo del VIC-II, situato all'indirizzo 53272 ($D018). Usando questi bit si può collocare lo schermo in uno dei blocchi a 16 Kbyte all'interno della finestra.

```
1040 REM ** SELEZIONA L'OFFSET TRA LO SCHERMO E LA BASE DELLA FINESTRA **
1050 SO = 1: REM SELEZIONA LO SCARTO NORMALE
1060 POKE 53272, (PEEK(53272) AND 15) OR 16*SO
```

In questo caso SO può assumere valori compresi tra 0 e 15 e l'istruzione PEEK(53272)AND 15 permette di conoscere in qualsiasi momento tale valore.
L'indirizzo in memoria corrispondente alla base dello schermo corrente può essere calcolato con la formula SC = VFIN + 1024*SO (base della finestra + offset), oppure con:

```
SC = 16384*(3—(PEEK(56576)AND3)) + 64—(PEEK(53272) AND 240)
```

Tuttavia c'è un piccolo problema relativo al movimento dello schermo: la RAM del colore non può essere spostata. Quindi, per avere uno schermo "alternativo" è necessario ricorrere a una piccola routine in codice macchina, capace di trasferire e richiamare la memoria di colore da un "buffer". Questa routine è già stata descritta a pag. 1398 per il programma degli schermi aggiuntivi sul Commodore 64.
Un secondo fattore da considerare è che, se si desidera visualizzare sul nuovo schermo, è necessario comunicare al sistema operativo (piuttosto che al chip video) la sua posizione in memoria, impiegando un'istruzione POKE (oppure STA) per indirizzare 648 ($0288), ossia il puntatore che dovrebbe contenere il byte alto dell'indirizzo di base della "memoria video". Se SC viene calcolato nel modo descritto in precedenza, occorre usare la seguente istruzione BASIC di indirizzamento:

```
POKE 648, INT(SC/256)
```

Per selezionare l'indirizzo di base del set dei caratteri o dello schermo in alta risoluzione, si usano le linee:

```
1070 REM ** SELEZIONA L'OFFSET DALLA BASE DELLA FINESTRA DI MC/HIRES **
1080 HO = 4: REM SELEZIONA L'OFFSET NORMALE
1090 POKE 53272, (PEEK(53272) AND 240) OR 2*HO
```

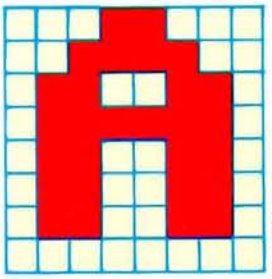
In teoria, HO può assumere qualsiasi valore compreso tra 0 e 7 ma in pratica esistono alcuni fattori che limitano la scelta. Per FIN equivalente a 1 o 3, HO non può avere il valore di 2 o 3 dato che ciò si verifica quando il VIC-II "vede" l'immagine normale dei caratteri ROM. Inoltre, valori elevati di HO pongono la parte superiore della memoria di alta risoluzione fuori dalla portata del VIC-II. L'indirizzo in memoria corrispondente alla base dello schermo corrente per la matrice di caratteri e per l'alta risoluzione viene calcolato come: MC = VFIN + 2048*HO (base della finestra + offset), oppure con l'espressione:

```
MC = 16384*(3—(PEEK(56576) AND 3)) + 1024*(PEEK(53272) AND 14)
```

Selezionando tali registri, è possibile spostare a piacimento l'intera memoria dell'immagine video.

Colori coordinati

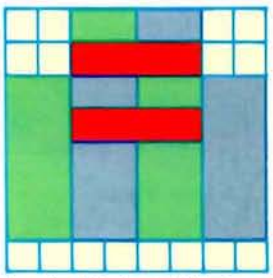
Nel modo di schermo "normale" del Commodore 64, ogni bit posto a 1, all'interno degli 8 byte che definiscono un carattere, viene visualizzato nel colore di primo piano corrente. I bit posti a 0 vengono invece visualizzati nel colore di sfondo dello schermo. Quando viene selezionato il modo Multicolour, i bit non vengono più interpretati singolarmente ma in coppia. Le quattro combinazioni possibili di una coppia di bit rappresentano il colore di primo piano, quello di sfondo e i due "multicolour" supplementari



LIZ DIXON

Il carattere 'A' nel modo di schermo "normale"

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0



Il carattere 'A' nel modo Multicolour

0 0	0 1	1 0	0 0
0 0	1 1	1 1	0 0
0 1	1 0	0 1	1 0
0 1	1 1	1 1	1 0
0 1	1 0	0 1	1 0
0 1	1 0	0 1	1 0
0 1	1 0	0 1	1 0
0 0	0 0	0 0	0 0

Chiave

01 Multicolour 1

10 Multicolour 2

11 Colore di primo piano

00 Colore di sfondo

"Knight Lore"

Dopo "Atic Atac" e "Sabre Wulf", divenuti dei classici, ecco un altro gioco della Ultimate

La stanza del mago



Lo Stivale



La Coppa



IAN McKINNELL

L'antro del licantropo
In "Knight Lore" lo scopo è quello di gettare nel calderone l'oggetto necessario affinché il protagonista, Sabre Man, non diventi un licantropo per sempre. Prima di partire in missione, quindi, occorre far visita allo stregone, che fa sapere di quale oggetto si tratti. Sebbene la collocazione degli oggetti venga cambiata, essi si trovano sempre nelle stesse stanze. Per arrivare allo Stivale, ad esempio, è necessario spostare i tavoli ed arrampicarsi. Per la Coppa, invece, il giocatore deve attendere che si apra il cancello giusto

La Ultimate si è sempre distinta per l'ottima qualità dei giochi destinati allo Spectrum Sinclair.
La ditta ha prodotto una serie di giochi a labirinto di stile arcade nei quali il giocatore deve aprirsi il cammino tra creature fantastiche che popolano le diverse stanze.
Contemporaneamente deve raccogliere oggetti magici e talismani che, riuniti assieme, gli permettono di concludere il gioco.
Esempi di questo tipo sono l'*Atic Atac* (vedere a pag. 856) e il *Sabre Wulf* (vedere a pag. 913).
Nonostante il successo riscosso da questi programmi, la Ultimate ha recentemente abbandonato la produzione di giochi d'avventura bidimensionali per passare al Knight Lore che, sebbene presenti alcune somiglianze con le precedenti creazioni, è un tipo di gioco completamente diverso.
Il protagonista è un cavaliere (si tratta di Sabre Man, l'eroe di molte precedenti avventure) che deve introdursi in un castello alla ricerca di una pozione magica che gli permetterà di non trasformarsi definitivamente in un licantropo.
Il giocatore ha 40 giorni e 40 notti di tempo per trovare la pozione: il trascorrere dei giorni viene visualizzato da una meridiana posta nell'angolo inferiore destro dello schermo.
Durante il giorno il protagonista è Sabre Man ma, quando sulla meridiana compare la luna, egli si trasforma tra atroci convulsioni (degne di Lon Chaney nella sua forma migliore) in un lupo mannaro.
Fortunatamente le possibilità di movimento del giocatore vengono inibite soltanto per la durata della trasformazione, ma in questo intervallo egli non può scansare eventuali oggetti in caduta.
Quando il gioco inizia, la prima cosa che colpisce il giocatore è l'ottima grafica tridimensionale degli scenari (già impiegata, anche se con minore effetto spettacolare, in un altro pacchetto per lo Spectrum: l'*Ant Attack*). Diversamente da quanto accade in *Ant Attack*, in *Knight Lore* non è possibile modificare il punto di vista dell'immagine, ma questa limitazione viene compensata dall'originale tecnica usata per costruire le stanze del labirinto.
Il *Knight Lore* non è un'avventura nel più puro stile arcade, ma piuttosto una serie di "puzzle" logici che il giocatore deve risolvere per poter avanzare nel labirinto. Molte delle stanze sono vuote ma altre contengono esseri che vanno evitati se si vuol andare avanti. Una stanza, ad esempio, è abitata da piccole creature simili a fantasmi che corrono sul pavimento: basta un loro "tocco" per far perdere una vita a Sabre Man!
Le prove da superare hanno vari livelli di difficoltà. Alcune sono relativamente semplici, altre diabolicamente complesse, altre ancora sono semplicemente false piste. Ad esempio, può accadere di entrare in una stanza piena di blocchi che sorreggono grandi globi spinosi: se si tenta di attraversare un blocco si corre il rischio di essere punti, quindi la soluzione è aggirare i blocchi. In altri casi i programmatori hanno giocato sulla distorsione della prospettiva, dovuta alla visione bidimensionale di uno spazio a tre dimensioni.
Spesso i cubi sembrano appoggiati al pavimento ma, in realtà, fluttuano nell'aria: se Sabre Man ne urta uno, viene trafitto dalle spine, perde una vita e deve ripartire da capo per attraversare la stanza.
Molte stanze contengono alcuni oggetti che il protagonista deve raccogliere come gemme, coppe e filtri magici.
Per impadronirsene, il giocatore deve avvicinare Sabre Man all'oggetto e farlo saltare sopra con un'abile mossa del joystick. L'oggetto catturato viene inserito in un inventario e visualizzato nella parte sinistra dello schermo. La manovra con il joystick è indispensabile per superare muraglie altrimenti invalicabili.
I movimenti di Sabre Man nelle quattro direzioni possono essere comandati sia tramite la tastiera sia con il joystick. I salti si ottengono premendo i tasti della terza fila o il pulsante di fuoco. Durante il gioco la corretta orientazione del protagonista è di capitale importanza: un salto nella direzione sbagliata può essere fatale.
Knight Lore è un gioco stimolante, degno della grande tradizione della Ultimate e capace di affascinare anche chi non ama i rompicapo.
La soluzione di molti dei suoi problemi richiede buona concentrazione e grande prontezza nell'uso dei tasti o del pulsante di fuoco.

Knight Lore: Per Spectrum da 48 Kbyte
Editori: Ashby Computer and Graphics Ltd (GB)
Autori: Ultimate Play the Game
Joystick: Opzionale
Formato: Cassetta

21497