DICEMBRE 1990
ANNO III

MSX DISK

N 22

- Nemesi
- Zanac 3
- Wonderboy
- The Games
- Turbocopy
- Notebook 2
- Finance Plus 5

giochi
news
utilità

£ 14.000

L'UNICA DISK MAGAZINE
DEDICATA ALLO STANDARD MSX

MSX DISK - REGISTRAZIONE N. 775 DEL 20/11/87
PRESSO IL TRIBUNALE DI MILANO - GRUPPO
EDITORIALE INTERNATIONAL EDUCATION
S.R.L., VIALE FAMAGOSTA 75, 20142 MILANO -
DIRETTORE RESPONSABILE: GRAHAM JO-
HNSON - DISTRIBUZIONE: MESSAGGERIE
PERIODICI S.p.A. ADERENTE A D.N., V.LE
FAMAGOSTA, 75 - MILANO - TEL. 84.67.545

SOMMARIO

2 Sommario – Sul disco
Caricamento – Avvertenze
3 Editoriale – Abbonamenti
4 Nemesi – Zanac 3
5 Wonderboy – The Games
6 Turbocopy - Notebook 2
8 Finance Plus 5
9 Il Pascal (Parte III[a])
12 Basic (Parte VII[a])
17 Computer & Programmazione (Parte V[a])
20 Dentro l'MSX (Parte IV[a])
23 Tendenze
24 Il libro del mese

SUL DISCO

1 Nemesi
2 Zanac 3
3 Wonderboy
4 The Games
5 Turbocopy
6 Notebook 2
7 Finance Plus 5

CARICAMENTO

A computer spento inserite il disco nel driver. Tenendo premuto il tasto CTRL accendete il computer e tenetelo inserito fino alla comparsa sul video del sommario. Per caricare un programma premete il numero corrispondente (dall'1 all'8). Il caricamento avverrà automaticamente.

AVVERTENZE

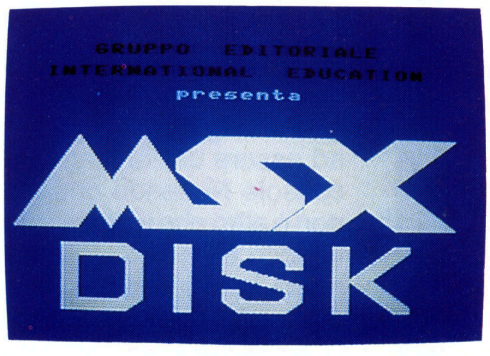
Questo disco è stato registrato con cura e con i più alti standard di qualità. Leggete con attenzione le istruzioni per il caricamento. Nel caso in cui, per una ragione qualsiasi, trovaste difficoltà nel caricare i programmi, telefonate alla nostra redazione al numero (02) 89502256 oppure spedite il disco al seguente indirizzo:

Gruppo Editoriale International Education srl - viale Famagosta, 75 - 20142 Milano.

Testeremo il prodotto e, nel caso, lo sostituiremo con uno nuovo senza aggiunta di costi supplementari.

EDITORIALE

La conclusione di un anno e l'inizio del successivo è tempo di festeggiamenti ma anche di consuntivi. Si guarda all'anno appena passato per capire quello che inizia, così vogliamo dare un'occhiata alla situazione attuale dell'universo computer.
Per questo, in questo numero troverete un'articolo dedicato alle tendenze del mercato, mentre ora vorrei soffermarmi su alcune considerazioni di cui è facile rendersi conto.
Il 1990 è un anno da ricordare nella storia dei computer: dopo la prematura scomparsa dello standard Msx, dopo l'estinzione dei computer ZX e con la dipartita degli ultimi Commodore 64 e derivati, quest'anno ha segnato la definitiva scomparsa dal mercato dei computer ad 8 bit.
Piano piano - rispetto ai ritmi informatici - gli home computer che hanno segnato un'epoca sono scomparsi dai negozi, specializzati o meno che siano. Sul mercato ora si trovano solo home computer a 16 bit che si riducono a due standard di base: i PC Compatibili e i 68000 come Commodore Amiga e Atari ST. Sono scomparsi tutti quegli home computer più o meno famosi che spuntavano come funghi nei primi anni 80. E stessa sorte subiranno le console per videogiochi: sul mercato già sono apparse quelle a 16 bit e vedrete che, per esempio, tra un anno esatto si parlerà poco del Sega Mastersystem a 8 bit che sarà soppiantato dal consanguineo Sega Megadrive a 16 bit.



E' forse con un po' di nostalgia che scrivo tutto questo, anche perché sono cresciuto informaticamente grazie al passaggio da un sistema all'altro, partendo dal primo mitico Vic 20 che molti, sono sicuro, ricorderanno con una lacrimuccia negli occhi.
Ma ora basta! Chiudiamo un'epoca e apriamo la nostra rivista. Buon lavoro & Buon divertimento!

LA REDAZIONE

ABBONAMENTI

Comunicato importante
da oggi potrete abbonarvi alla rivista MSX DISK e riceverla comodamente a casa semplicemente sottoscrivendo uno speciale abbonamento per 10 numeri allo specialissimo prezzo di Lit. 128.000 invece di Lit. 140.000. Potrete così assicurarvi la vostra copia e risparmiare ben Lit. 12.000.

Desidero abbonarmi alla rivista MSX DISK allo speciale prezzo di Lit. 128.000 anziché Lit. 140.000 per 10 numeri.

COGNOME _____ NOME _____

VIA _____

CAP _____ CITTA' _____ PROV. _____

Allego assegno ☐ vaglia postale ☐ intestato a **Gruppo Editoriale International Education.**

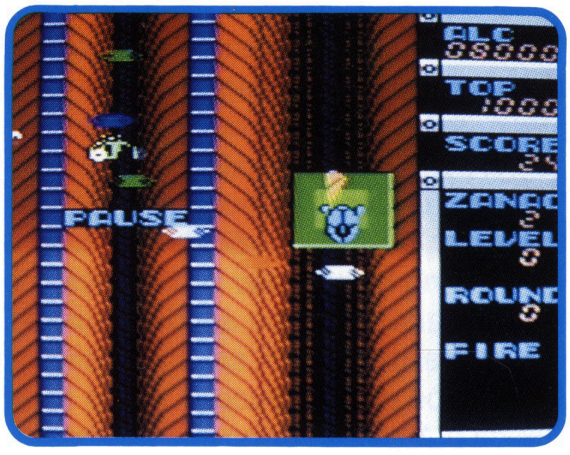
Ritagliare e spedire a Gruppo Editoriale International Education – Viale Famagosta 75 – 20142 Milano.

NEMESI

Questo gioco è il prosieguo di uno dei più famosi giochi per Msx, il primo ad aver inaugurato i sistemi Msx 2, ed il primo ad essere uscito, con un altro nome, per la console Nintendo che, come pochi sanno, non è altro che un Msx 2 con qualcosa di più.
Il gioco è un classico shoot'em'up, cioè un gioco in cui si deve sparare a destra e a manca cercando di raccogliere bonus e cose del genere. Diviso in più differenti livelli, Nemesi è un gioco a scorrimento orizzontale in cui dovrete guidare la solita astronave verso la solita base finale. La grafica e ottima e fate attenzione ai dettagli.
Distruggendo gli stormi di astronavi aliene o le più letali postazioni aliene, sullo schermo compariranno delle nuvolette bonus che dovrete raccogliere per potenziare il vostro armamento. Una volta caricato il gioco vi verrà posto un menù principale di cui dovrete scegliere un'opzione per poter iniziare il gioco. La selezione si effettua muovendo il cursore con gli appositi tasti - quelli con le frecce! - e poi premendo la barra spaziatrice. In alternativa potete anche usare il joystick, come sempre inserito nella porta numero 1.
Durante la visualizzazione delle istruzioni (in inglese) e la dimostrazione, che si avviano automaticamente dopo alcuni secondi di permanenza nello schermo di presentazione, potrete tornare al menù principale semplicemente premendo lo spazio.

COMANDI
Tasti: [CURSORI] = Movimenti
[SPAZIO] = Inizio gioco / Fuoco
[F1] = Attiva/Disattiva pausa
Joystick in porta 1

ZANAC 3



Il pianeta Comna è un luogo ameno dove la vita scorre tranquilla come sulla Terra, è ora sotto la minaccia di un attacco da parte degli esseri abitanti il sottospazio di Trixab che vogliono distruggere ogni forma di vita sul pianeta. Per salvare Comna hai lanciato nello spazio una navetta, un nuovo prototipo da combattimento iperspaziale denominato ZANAC. L'intera galassia è ora in trepida attesa del mortale duello che dovrai affrontare contro i crudeli Trixabiani. Il tuo obiettivo sarà la distruzione totale delle flotte e degli insediamenti nemici su Comna fino a raggiungere e distruggere la base del potere centrale. Per questo avrai bisogno di tutto il tuo coraggio, di riflessi prontissimi e della massima determinazione in ogni tua azione.
Il computer di bordo della tua nave Zanac è già programmato per condurti alla base di potere centrale ma la rotta passa per le installazioni e postazioni nemiche e così dovrai distruggerle tutte e difenderti dall'attacco degli stormi di caccia stellari nemici.
Distruggendo le postazioni nemiche e gli stormi di navi aliene sullo schermo compariranno dei bonus che se raccolti ti forniranno di potenti armi e nuova energia per continuare la tua missione. Inizialmente potrai sparare in avanti e di lato, a seconda della direzione laterale in cui ti muovi. Nella parte destra dello schermo troverai il punteggio, il record del giorno e tutte le informazioni fornite dal computer di bordo.

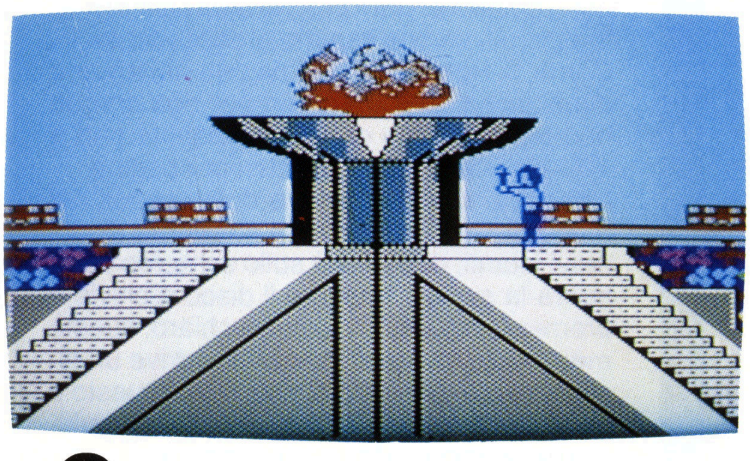
COMANDI
Tasti: [CURSORI] = Movimenti
[SPAZIO] = Inizio gioco / Fuoco
[STOP] = Attiva/Disattiva Pausa
Joystick in porta 1

WONDERBOY

E' tempo di vacanze natalizie e Ciccio decide di andare in Kenia dove, in un caldo e assolato pomeriggio di Dicembre (?), decide di mollare tutto e di addentrarsi all'interno della foresta tropicale alla ricerca della vita selvaggia. Qualcuno però gli ha suggerito di essere prudente e portarsi con se qualche arma, ma Ciccio, che sente il richiamo della foresta sempre più forte, non vuole saperne di diavolerie moderne e decide di portare con se un primitivo boomerang. Appena addentratosi nella foresta, il nostro eroe si rende conto che rimanere in vita sarà tutt'altro che facile e rimpiange la civiltà che ora vuole riabbracciare. Ma la strada verso la città sarà dura e gli ostacoli che si troverà davanti saranno così tanti che non avrà il tempo per tirare il fiato. Ciccio potrà solo contare sul suo rudimentale boomerang, che userà in continuazione, e sulle "sorprese" che spunteranno dalle strane uova di dinosauro che incontrerà lungo la strada. Alcune lo renderanno invulnerabile al punto tale che sarà in grado di superare incolume ostacoli notevoli, quali velenosissimi serpenti. Altre sorprese gli regaleranno uno stupendo skateboard che non si capisce cosi ci faccia lì. Attento pero! La vulnerabilità durerà poco e potrai perdere lo skateboard molto facilmente. Un consiglio: dai modo a Ciccio di recuperare le forze raccogliendo la frutta sparsa per il sentiero.

COMANDI
Tasti: [CURSORI] = Movimenti
[SPAZIO] = Inizio gioco / Salto & Lancio Boomerang
Joystick in porta 2

THE GAMES



Questo programma offre la possibilità di competere in sette diverse specialità dei giochi olimpici invernali. Una volta completato il caricamento del programma sullo schermo apparirà la presentazione iniziale e, premendo la barra spaziatrice, si potrà accedere al menù principale composto da otto opzioni. La prima opzione permette di gareggiare in tutte le specialità, la seconda solo in alcune specialità a scelta mentre la terza in una sola specialità. La quarta opzione è simile alla terza, ma invece di competere permette di fare pratica nella specialità scelta. La quinta opzione fa passare a due identici menù secondari posti in sequenza e che permettono di scegliere i controlli per il primo e per il secondo giocatore. Questi menù secondari permettono di assegnare ad ogni giocatore l'uso di uno dei due joystick o della tastiera, e anche di scegliere i tasti per controllare il gioco. La sesta opzione del menù principale permette di vedere i record relativi ad ognuna delle sette specialità con il nome dell'autore. La settima opzione fa vedere la cerimonia di apertura dei giochi olimpici mentre con l'ottava opzione è possibile vedere quella di chiusura. Una volta scelto l'evento o gli eventi in cui competere è necessario inserire il nome di ogni giocatore - da uno a otto - e la relativa nazionalità. Per scegliere la nazionalità si deve spostare il cursore sulla bandiera scelta e premere il tasto di fuoco, dopodiché si assisterà all'esecuzione del relativo inno nazionale.
Selezionando la tastiera come controllo senza ridefinire i tasti, dovrete usare i seguenti tasti per controllare il gioco:

COMANDI
Joystick in porta 1 e 2
TASTI: [Q], [A], [E], [R] e [Z] = per un giocatore
[P], [L], [U] e [,] = per l'altro giocatore

TURBOCOPY

Nell'ormai ricca serie di copiatori che vi abbiamo proposto sino ad ora non poteva certamente mancare un programma che permetta di copiare i dischi molto velocemente. TurboCopy colma questo vuoto offrendo la possibilità di copiare dischetti velocissimamente. Normalmente per copiare un dischetto settore per settore utilizzando un solo drive richiede una notevole dose di pazienza, ma con TurboCopy rimarrete estasiati.
Il programma è destinato all'uso esclusivamente con macchine Msx 2 poichè sfrutta una gestione particolare dei dati, possibile solo su queste macchine.
Il programma sfrutta un metodo decisamente particolare ed insolito. I dati letti finiscono interamente nella memoria video (VRAM) e nella memoria video espansa della macchina che è decisamente più veloce della normale Ram. Oltre a questo, il programma compatta velocemente i dati consentendo di leggere e registrare i dati letti con la massima velocità.
Una volta caricato il programma dopo averlo selezionato nel menù programmi di Msx Disk, sullo schermo compariranno alcuni messaggi. A questo punto dovrete inserire il disco originale e premere lo spazio. Dopo alcuni istanti comparirà un nuovo messaggio e dovrete premere nuovamente lo spazio senza sostituire il disco. Da questo momento in poi il programma leggerà la prima parte di dati fino a quando non comparirà un nuovo messaggio.
Giunti a questo nuovo messaggio dovrete sostituire il disco originale con quello di destinazione che avrete opportunamente formattato prima di cominciare ad usare il copiatore. Una volta inserito il disco di destinazione - che deve essere formattato a 360k o 720k a seconda della formattazione del disco originale - dovrete premere di nuovo lo spazio, dopodichè il programma riverserà i dati letti sul nuovo disco. Una volta terminata questa operazione, sullo schermo comparirà un nuovo messaggio che vi inviterà ad inserire il disco originale e a premere nuovamente lo spazio. Continuate alternando disco originale e copia e premendo lo spazio fino a che non comparirà il messagio che indica la fine della copia.
Attenzione! Ricordate di proteggere sempre da scrittura il disco originale poichè eventuali errori da parte vostra potrebbero risultare irrecuperabili. Pur disponendo di due drive non potrete sfruttare TurboCopy con entrambe poichè questa versione del programma non è fatta per sistemi con doppio disk drive.

NOTEBOOK 2

Nel numero 19 di Msx Disk vi abbiamo proposto l'utility Notebook, in pratica una specie di archivio per delle paginate di testo. Proprio come il predecessore, Notebook 2 è un utility di archiviazione che permette di creare un vero e proprio blocco notes, Notebook in inglese, che permette di archiviare su disco delle pagine di testo.
Ovviamente il programma non è uguale a quello già propostovi, ma presenta numerose migliorie, soprattutto per quanto riguarda quei piccoli bug che rendevano non perfettamente funzionante il primo programma. Per esempio, nel primo Notebook non era possibile modificare automaticamente le pagine di uno stesso testo mentre questo ora e possibile, poi il programma risulterà un tantino più veloce rispetto al precedente.
Come in Notebook 1, le pagine hanno una dimensione di 40 caratteri per riga con un numero massimo di rige per pagina pari a 120, cioè circa 4800 caratteri per pagina.
Una volta scelta l'opzione di Notebook nel menù di Msx Disk dovrete attendere alcuni brevi istanti, dopodiché sullo schermo comparirà la scritta: "Inserire il disco archivio e premere un tasto...". Come è chiaro, questo messaggio invita ad inserire nel drive un disco diverso da quello di Msx Disk. Questo è necessario in quanto non ci sarebbe sufficiente spazio sul disco per ospitare gli archivi di Notebook. Per rendere più efficiente il sistema potete copiare il programma - che sul nostro disco è chiamato 5.BAS - direttamente in un disco archivio che userete solo per questo scopo. Rinominando il file come AUTOEXEC.BAS potrete far partire automaticamente il programma di gestione ogni volta che inserite il disco archivio e accendete il computer.

segue

NOTEBOOK 2

Dopo la pressione di un tasto qualsiasi una volta inserito il disco archivio, sullo schermo comparirà il menù principale di Notebook che potete vedere nella foto.
Il programma offre 7 diverse opzioni selezionabili premendo il tasto numerico corrispondente e quindi il tasto di invio.
Le opzioni disponibili sono:

1 - INSERIMENTO

Selezionando questa opzione sullo schermo apparirà la richiesta di una chiave che identificherà i diversi fogli del blocco.
Se usate più volte la stessa chiave i nuovi fogli aggiunti diventeranno il secondo, il terzo, il quarto e così via di fogli appartenenti a quel soggetto.
Nuovi fogli con la stessa chiave vengono inseriti ogni volta che superate lo spazio destinato ad una singola pagina. In questo caso il computer archivia la pagina precedente e vi porta direttamente alla nuova pagina.
Dopo aver inserito la chiave potete passare alla fase di inserimento vera e propria.
Per terminare l'inserimento della pagina premete il tasto di Escape [ESC] che riporta l'esecuzione alla richiesta della chiave.
Inserendo una chiave nulla l'esecuzione ritorna al menù principale.

2 - MODIFICA

Questa opzione, come la precedente, vi richiede la chiave della pagina da modificare.
Se esistono più pagine con la stessa chiave vi verrà chiesto il numero di pagina, da 1 a N dove N è il numero di pagine con la stessa chiave. Dopo aver raggiunto la pagina la fase di editing sarà in tutto e per tutto uguale a quella dell'inserimento.
Come nella prima opzione, dovrete usare il tasto [ESC] per terminare la modifica.

3 - ELIMINAZIONE

Anche qui dovrete indicare la chiave di ricerca. Trovata la pagina desiderata il computer vi chiederà la conferma dell'operazione di cancellazione della pagina. Dovrete necessariamente rispondere inserendo [S] altrimenti la pagina non verrà cancellata.

4 - ANALISI

Questa opzione vi porta ad un sottomenù con tre opzioni:

LISTA PAGINE
che mostra una lista delle pagine presenti chiave per chiave.

ANALISI PAGINA
che permette di visualizzare, riga per riga e velocemente, una o più pagine di una chiave data.

MENU' PRINCIPALE
per tornare al menù precedente.

5 - STAMPA

Come nel caso precedente, tramite questa opzione si accede ad un menù secondario con tre opzioni:

STAMPA PAGINA
che stampa una o più pagine con la stessa chiave.

INSTALLAZIONE STAMPANTE
che permette di formattare la stampa.

MENU' PRINCIPALE
che riporta l'esecuzione al menù principale.

6 - CAMBIO DISCO

Questa opzione serve per cambiare disco archivio durante l'elaborazione con anche la possibilità di formattare il nuovo disco archivio.

7 - FINE LAVORO
Questa opzione permette di abbandonare l'uso di Notebook.
Dopo averla selezionata dovrete confermare se abbandonare il programma inserendo [S] o [N].
Poichè le pagine gestite da Notebook 2 sono composte da 4800 caratteri in un disco formattato a 360 kilobyte potrete inserire un massimo di 75 diverse pagine, valore che raddoppia usando dischetti formattati a 720 kilobyte.

FINANCE PLUS 5

Come già detto in precedenza, scopo di questi programmi è di fornire un aiuto nella gestione delle proprie finanze e offrire uno spunto didattico ai principianti.
Il disco contiene sia la versione compilata e funzionante del programma, sia il file sorgente - con estensione .BAS - già pronto per essere usato o analizzato tramite un editor o qualsiasi interprete (come il GWBASIC) o qualsiasi compilatore (QUICK BASIC o TURBO BASIC).
Per utilizzare il programma già compilato sarà sufficiente digitare FINANCE5 e premere il tasto di invio ([RETURN] o [ENTER]).
Il file FINANCE5.BAS è il file sorgente pronto per voi. Questa volta il programma è dedicato al metodo di deprezzamento accelerato e alla suddivisione proporzionale.
Il programma è, come le volte scorse, diviso in tre parti e offre un menù principale con le seguenti opzioni:

1 - TASSO INTERNO DI RENDIMENTO;
2 - TASSO DI RENDIMENTO DELLA GESTIONE FINANZIARIA;
3 - RITORNO AL DOS.

Nel file sorgente potrete trovare che la prima opzione viene espletata dalla parte che va dalla linea 1000 alla linea 1590 mentre l'analisi degli investimenti comuni è compresa tra la linea 2000 e la linea 2990. Come sempre, dalla linea 3000 in poi trovano posto le routine di servizio.

TASSO INTERNO DI RENDIMENTO

Il tasso interno di rendimento è il tasso con cui la somma dei ricavi compensa l'investimento iniziale. Il programma calcola questo tasso applicando un metodo di ricerca per bisezione. Per usare il programma è necessario inserire nell'ordine i seguenti dati: ammontare dell'investimento iniziale, termine dell'investimento (in anni), ammontare degli introiti per ogni anno. Le uscite - in questo caso i capitali investiti -vanno inserite come numeri negativi. Per ritornare al menù principale dovrete inserire zero come investimento iniziale. Questa routine può essere usata anche per calcolare la resa alla scadenza di un'obbligazione, inserendo il prezzo dell'obbligazione come ammontare dell'investimento iniziale, il numero di anni alla scadenza come termine dell'investimento, l'ammontare delle cedole da ricevere per ogni anno come gli introiti relativi a quegli anni (inserire per ogni anno l'ammontare totale), e, per quanto riguarda l'ultimo anno, inserire la somma del valore delle cedole ricevute e del valore a scadenza dell'obbligazione. Il valore calcolato dal programma è la resa a scadenza dell'obbligazione.
La ricerca col metodo di bisezione alle linee 1320-1540 trova tassi di rendimento compresi tra 0% e 99%. Se questo intervallo non è sufficientemente ampio per le vostre necessità, cambiate i valori iniziali della variabile L alla linea 1330 e della variabile H alla linea 1340. Queste due variabili contengono i limiti inferiore e superiore dell'intervallo di ricerca. Assicuratevi che quando la linea 1370 viene eseguita per la prima volta, il valore di (L + H)/2 non sia zero, perché in questo caso l'algoritmo verrebbe interrotto anzitempo da un errore DIVISION BY ZERO.
ESEMPIO: Il signor Rossi ha l'opportunità di investire in una certa impresa. L'investimento iniziale ammonta a 10.000.000 di lire, con introiti per ognuno dei primi tre anni rispettivamente di 4.000.000, 5.000.000 e 3.000.000 di lire. Il tasso di rendimento richiesto è del 15%. E' opportuno che il signor Rossi accetti di investire?
RISPOSTA: No. Il tasso di rendimento è del 10.1331%.

TASSO DI RENDIMENTO DELLA GESTIONE FINANZIARIA

Questo indicatore differisce dal precedente per molti aspetti. Nel caso di certi investimenti, ed in particolare di quelli riguardanti beni immobili, il primo fornisce una stima più realistica del secondo. Il programma considera solo entrate già depurate dalle tasse e dal finanziamento e ignora il fatto che siano disponibili alle fonti di finanziamento. Per usare il programma, inserite la durata dell'investimento - in anni - e il tasso di investimento liquido. Quest'ultimo non è altro che il tasso al quale i fondi possono essere investiti ad una certa percentuale al netto delle tasse e ritirati quando lo si desideri, per esempio nel caso di un libretto di risparmio. Potete anche inserire un tasso di investimento fisso 'garantito', e cioè tale che il rendimento dell'investimento assommi sicuramente almeno ad una certa cifra minima prestabilita.
Un investimento del tipo di cui stiamo parlando, potrebbe essere per esempio un progetto immobiliare o qualche altro investimento fisso di rischio comparabile a percentuali al netto delle tasse superiori al tasso liquido, come certificati di deposito o buoni del Tesoro. Il tasso di investimento fisso dovrebbe avere un ammontare minimo che possa essere investito. Inserire anche questo ammontare.
Il programma indicherà i punti in corrispondenza dei quali dovreste investire dei capitali nell'investimento liquido e in quello fisso, l'effettivo investimento che potete fare (la differenza tra questo ammontare e l'investimento originale deve essere investito al tasso fisso all'inizio del primo anno), il rendimento effettivo dell'investimento, il tasso al quale il rendimento totale effettivo sconta l'investimento iniziale effettivo.
ESEMPIO: Il signor Bianchi ha pianificato la costruzione di un gruppo di appartamenti. I termini sono: 10.000.000 di lire di acconto da pagare subito, più il pagamento di 50.000.000 di lire da farsi nel corso dei due anni seguenti. Alla fine del terzo e del quinto anno Bianchi può spettarsi di ricevere 30.000.000 di lire dal suo investimento. Egli prevede di ristrutturare l'edificio durante il quarto anno, ad un costo stimato in 20.000.000 di lire. Infine, al sesto anno, egli prevede di vendere l'edificio per 250.000.000 di lire. Il tasso di investimento liquido disponibile è del 5%, e ad un investimento fisso di un minimo di 10.000.000 di lire corrisponderà un rendimento sicuro di almeno il 10%. Qual'è il tasso di rendimento della gestione finanziaria nel caso dell'investimento di Bianchi?
RISPOSTA: 19.348% - il tasso di rendimento interno in questo caso sarebbe del 25.2%.

IL PASCAL (Parte III[a])

Dopo aver parlato del "vocabolario" nell'articolo del numero scorso, questa volta cominceremo a vedere la struttura generale di un programma Pascal:

le Risorse e l'Algoritmo,

la Struttura di un Modulo,

la descrizione delle Risorse e dell'Algoritmo,

fino ai primi, diversi "tipi".

RISORSE ED ALGORITMO

Supponiamo di voler descrivere un programma che trasferisce un testo riga per riga da un dispositivo di ingresso (un terminale) ad un dispositivo di uscita (una stampante).
Seguendo l'approccio metodologico basato sulla decomposizione del problema per raffinamenti successivi, al primo livello di dettaglio possono essere individuate tre risorse su cui basare l'algoritmo risolutivo.
Le tre risorse individuate sono il dato su cui si opera, ovvero la singola riga di testo, e due moduli che descrivono sotto-algoritmi non esplicitati, ma che eseguono rispettivamente le funzioni di mlettura e di scrittura di una riga di testo.
L'algortimo risolutore del problema consiste in un ciclo terminante su una condizione di fine testo che sequenzialmente legge una riga e la scrive.

```
‹descrizione di "riga"›
‹descrizione di "leggi"›        SPECIFICA RISORSE
‹descrizione di "scrivi"›

‹cicla fino a fine testo›
‹leggi una riga›                SPECIFICA ALGORITMO
‹scrivi una riga›
```

A un successivo livello di dettaglio siamo tuttavia interessati a risolvere i sottoproblemi di lettura e scrittura.
Le due risorse modulo devono essere ulteriormente specificate, ovvero deve essere indicata la loro realizzazione.
Anche in questo caso il sottoproblema ‹leggi una riga› può essere risolto individuando risorse sulle quali opererà l'algoritmo descrittivo.
In particolare possiamo individuare il singolo carattere come risorsa e un modulo di più basso livello che esegue la funzione di lettura del singolo carattere.

L'elaborazione di tale modulo consisterà nell'iterazione di letture di singoli caratteri fino a incontrare un carattere di "fine linea".

```
‹descrizione di carattere›
‹descrizione di leggicarattere›     SPECIFICA RISORSE

‹cicla fino a carattere "fine linea"›
‹leggi un carattere›                SPECIFICA ALGORITMO
```

Come si vede la descrizione di un modulo è identica a ogni livello di dettaglio considerato e si compone della specifica delle risorse usate e della elaborazione delle stesse.

STRUTTURA DI UN MODULO

La struttura testuale di un programma Pascal riflette da vicino l'approccio sopra descritto.
Un programma Pascal è descritto da una carta sintattica dove un ‹program header› specifica il nome del programma e i suoi rapporti con l'esterno, tipicamente i canali di I/O, specificati da identificatori; e poi un ‹blocco› che è la forma sintattica che include gli aspetti di specifica di risorse e di elaborazione sia a livello di programma principale sia di modulo.
Con le conoscenze sintattiche fin qui acquisite, il programma dell'esempio può essere parzialmente specificato nel modo seguente:

```
PROGRAM trasferisci(input, output);
‹descrizione delle risorse›
‹specifica algoritmo›
```

DESCRIZIONE DELLE RISORSE

Tutte le risorse usate nella parte algoritmica di ogni modulo devono essere precedentemente specificate. A ogni risorsa verrà associato un identificatore che la denota e sarà usato per riferirla nella parte algoritmica.
Codificando l'esempio precedente in Pascal, la risorsa ‹riga› sarà descritta da una dichiarazione di ‹variabile›, ovvero di un oggetto che conterrà di volta in volta la riga di testo letta.
I due moduli per la lettura e la scrittura saranno descritti da due dichiarazioni di ‹procedura›, che sono le unità di programma presenti in Pascal.
In aggiunta il Pascal permette di associare identificatori ad altre entità, considerate anche queste risorse, sebbene di diversa natura.

IL PASCAL (Parte III[a])

In particolare è possibile definire sinonimi per tipi, costanti, etichette ("label").
Nel nostro esempio la strutura del dato ‹riga› potrebbe essere specificata in una definizione di tipo e separata testualmente dalla dichiarazione della variabile.
La definizione di tipo introduce un sinonimo per un modello di dato, mentre la dichiarazione di variabile esplicita una istanziazione di quel prototipo. Evidentemente più istanze di uno stesso modello possono essere esplicitate.
Ad esempio possiamo avere:

```
TYPE complesso = ‹tipo›;
VAR x, y, z: complesso;
```

L'uso dei sinonimi per le descrizioni dei tipi di dati ha molti vantaggi.
Innanzitutto la scelta appropriata dei nomi può migliorare la leggibilità del programma.
Inoltre un cambio nella definizione di un tipo non richiede la modifica di tutte le dichiarazioni delle variabili di quel tipo, eventualmente sparse nel programma (località della modifica).
Infine, come vedremo successivamente, la definizione di modelli di strutture di dati complesse è scritta solo una volta e non ripetuta in tutte le dichiarazioni di variabili o definizioni di altri tipi che la usano, riducendo la possibilità di errori.
Anche l'uso di nomi simbolici per valori costanti migliora la leggibilità e la manutenibilità del progrmama.
In particolare, una futura esigenza di cambiare il valore necessiterà una modifica solamente nella definizione piuttosto che in ogni uso della costante.
Alcuni esempi corretti di definizioni di costanti simboliche sono:

```
CONST max = + 527;
pigreco = 3.14;
nome = 'PIPPO';
min = - max;
```

Considerando al momento solo procedure senza parametri i due moduli dell'esempio precedente possono essere dichiarati come

```
PROCEDURE leggi;
.....
PROCEDURE scrivi;
```

Riprenderemo successivamente la trattazione specifica circa le unità programma del Pascal (comprese le funzioni) e le "label".
Il programma dell'esempio può ora essere ulteriormente specificato in questo modo, lasciando ancora indefiniti i blocchi delle procedure "leggi" e "scrivi":

```
PROGRAMM trasferisci (input, output);
TYPE linea = ‹tipo›;
VAR riga: linea;
PROCEDURE leggi;
‹blocco›;
PROCEDURE scrivi;
‹blocco›;
‹specifica algoritmo›
```

DESCRIZIONE DELL'ALGORITMO

La parte algoritmica di un blocco, cioè la categoria sintattica ‹specifica algoritmo›, è descritta semplicemente da uno statement composto.
Uno

STATEMENT COMPOSTO

è una sequenza di STATEMENT racchiusa da due parole riservate di inizio e fine.
Ogni statement è separato dal successivo da un separatore, il simbolo speciale ";".
Ogni statement specificato può essere una qualunque delle frasi eseguibili del linguaggio, ovvero semplici statement operativi o strutture di controllo.
Due osservazioni sono da fare sulla natura sintattica del linguaggio.
La prima riguarda la possiblità di usare uno statement compossto addove c'è uno statement.
In altre parole uno statement può essere una frase operativa, una struttura di controllo oppure una sequenza di statements.
La seconda riguarda l'uso del simbolo ";" come separatore di statement.
Questo conduce alla possibilità di avere più ";" successivi (indicanti la presenza di uno statement vuoto) o nessun ";" prima di un END.

UN ESEMPIO PARZIALMENTE COMPLETO

A questo punto abbiamo un programma parzialmente specificato dal punto di vista sintattico, che mette in mostra la sua struttura complessiva.
E' da enfatizzare il fatto che tale testo è stato ottenuto applicando successivamente le carte sintattiche introdotte.

IL PASCAL (Parte IIIª)

Solamente la parte algoritmica del programma principale è stata codificata completamente ed è composta di 6 statements.
Dal punto di vista sintattico è stata ottenuta attraversando per 6 volte la scatola ‹statement›.

```
PROGRAM trasferisci (input, output);
TYPE linea = ‹tipo›;
VAR riga: linea;
PROCEDURE leggi;
‹blocco›
PROCEDURE scrivi;
‹blocco›
BEGIN
reset(input);
rewrite(output);
WHILE NOT eof (input) DO
BEGIN
leggi;
“variabile “riga” riempita“”
scrivi;
END
END.
```

Ma cerchiamo ora di definire i tipi nel modo usato dal Pascal, e cominciamo a parlare dei diversi “tipi semplici”.

TIPI IN PASCAL

Un TIPO di dato è definito come una classe di valori ed un insieme di operazioni permesse su tali elementi.
Ad esempio, possiamo considerare la classe di numeri interi e le usuali operazioni aritmetiche come un tipo di dato denotato dal nome

INTEGER.

Una variabile che può assumere valori numerici interi sarà detta di tipo INTEGER.
Nel linguaggio Pascal a ogni variabile e costante deve essere associato un unico tipo di dato e questa associazione è stabilita

STATICAMENTE

al momento della dichiarazione o definizione dell'oggetto.
Ad esempio:

```
CONST max = 120;
VAR peso: integer;
```

sia la costante simbolica sia la variabile sono oggetti di tipo INTEGER.

Ogni tipo è classificato come:

TIPO SEMPLICE
è definito come insieme di valori elementari.

TIPO STRUTTURATO
i suoi valori sono strutture di valori elementari ed è definito specificando tali componenti e il metodo di aggregazione.

TIPO POINTER
usato per definire strutture dati dinamiche, variabili cioè in grandezza e forma durante l'esecuzione.

TIPI SEMPLICI

Un TIPO SEMPLICE è caratterizzato da un insieme di valori elementari.
Questi valori rappresentano atomi che non possono essere ulteriormente scomposti.
Tutto il calcolo esprimibile in Pascal si basa su questi semplici valori atomici.
In particolare per utilità del programmatore e per maggiore astrazione il linguaggio permette di definire sinonimi di tipi precedentemente definiti.
Analogamente alle costanti simboliche un modello di tipo di dato può essere denotato da più nomi.

Ad esempio

```
TYPE complesso = ‹tipo›;
vettore = complesso;
coordinata = complesso;
```

(continua)

BASIC (Parte VII[a])

Gli array consentono di trattare con un unico nome un insieme di variabili tra loro correlate, come dati misurati, elenchi di nomi, contenuto delle caselle di una scacchiera.
Sono composti da elementi numerati, identificabili con un indice, ciascuno dei quali è una normale variabile numerica o stringa.

ARRAY O VARIABILI CON INDICI

Riassumiamo quanto abbiamo finora visto a proposito delle variabili. In BASIC, si dividono in due TIPI fondamentali:

VARIABILI NUMERICHE, che contengono un valore numerico;
VARIABILI STRINGA, che contengono sequenze di caratteri.

Le variabili numeriche si distinguono poi a seconda della PRECISIONE, cioè del numero massimo di CIFRE SIGNIFICATIVE (e possono anche essere utilizzate come VARIABILI LOGICHE), con la convenzione che il valore zero corrisponde a FALSO ed ogni altro valore a VERO. Spesso, però, risulterebbe comodo associare un gruppo di variabili simili: questo non è possibile con le variabili che abbiamo visto finora.
Ci spieghiamo meglio con un esempio.
Supponiamo di essere in un laboratorio di fisica, dove occorre eseguire una serie di misure sperimentali. Sui dati di queste misure dobbiamo poi effettuare diverse analisi statistiche: calcolare il valore medio, il minimo, il massimo, lo scarto quadratico medio, etc.. E' quindi utile poter raccogliere i dati in una serie di variabili, con le quali eseguire poi i calcoli. Se i dati sono pochi, per esempio 5, potremmo scrivere:

```
100 INPUT D1
110 INPUT D2
120 INPUT D3
130 INPUT D4
140 INPUT D5
```

e poi usare le cinque variabili D1, D2, D3, D4 e D5 per i calcoli.
Nel nostro caso, il valore medio sarebbe:

```
200 VM = (D1 + D2 + D3 + D4 + D5) / 5
```

Gli altri calcoli sarebbero più complessi, ma comunque sempre fattibili.
Ma se i dati fossere 100, o 2000? Sarebbe impensabile usare altrettante variabili per contenerli: la scrittura delle operazioni diverrebbe semplicemente impossibile. Per nostra fortuna, però, il BASIC ci fornisce uno strumento per risolvere il problema: l'ARRAY (letteralmente "schiera", ma è meglio usare il termine originale).
Un array, o VARIABILE CON INDICI, non è altro che un insieme ordinato di variabili semplici (o SCALARI), raccolte sotto uno stesso nome. Nel caso più semplice, quello dell'array ad un solo indice (o VETTORE), possiamo pensare ad una fila di caselle numerate, ciascuna delle quali (ELEMENTO dell'array) è una variabile semplice. Nel caso del nostro esempio, perciò, invece delle cinque variabili distinte D1, D2, D3, D4 e D5, avremmo i cinque elemnti D(1), D(2), D(3), D(4), D(5) dell'array D.
Dove sta la differenza?
Lo vediamo subito con questo esempio:

```
100 FOR I = 1 TO 5
110 PRINT D(I)
120 NEXT I
```

Questo programmino stampa i valori delle cinque variabili. In dettaglio, la linea 100 dice di ripetere cinque volte quanto segue, fino alla linea 120 (NEXT), con il valore di I che parte da uno e aumenta di uno ad ogni passaggio (ciclo).
La linea 110 stampa il valore dell'elemento numero I dell'array D, cioè della Iesima casella dell'array. Al primo giro, I vale uno, perciò la linea 110 equivale a PRINT D(1), cioè stampa il valore del primo elemento di D. Al secondo giro, I vale due e viene stampato il valore contenuto nel secondo elemento, e così via.
Lo stesso programmi di tre istruzioni, però, potrebbe stampare 2000 valori, semplicemente sostituendo la linea 100 con FOR I = 1 TO 2000.
Gli array non sono limitati ad un indice (o una dimensione); si possono avere array a due indici (MATRICI rettangolari), che possiamo pensare come scacchiere o fogli quadrettati (divisi in righe e colonne), in cui ogni casella contiene una variabile semplce. Si possono avere anche array a tre o più indici: il BASIC non pone, di solito, limiti al numero delle dimensioni.
Comunque, gli array di gran lunga più usati sono quelli ad una e due DIMENSIONI (indici), indicati comunemente come VETTORI e MATRICI.
Anche gli array hanno un tipo è il tipo degli elementi in essi contenuti, e si indica con i soliti suffissi.
Per esempio:

```
A(5)
```

E' il quinto elemento dell'array numerico (reale) A.

```
XY%(40)
```

E' il quarantesimo elemento dell'array numerico (intero) XY%.

```
NOME$(2,3)
```

E' il terzo elemento della seconda riga dell'array stringa NOME$.

I primi due esempi sono ad una dimensione (vettori), il terzo a due (matrice).

DIMENSIONAMENTO DEGLI ARRAY

In BASIC non occorre DICHIARARE le variabili, cioè avvertire in anticipo il calcolatore di quali variabili il programma usa e di che tipo sono. Questo può sembrare, come detto, un vantaggio, ma porta in realtà ad inconvenienti dal punto di vista della chiarezza e della facilità di messa a punto (debug) del programma. Inoltre, l'efficienza e la velocità di esecuzione ne risultano diminuite.
Gli array, comunque, rappresentano l'unica eccezione a questa regola: poiché possono occupare ampie zone di memoria, occorre che il calcolatore sappia in anticipo quanta memoria "riservare" per ciascuno di essi.
Questo si chiama DIMENSIONARE gli array, cioè dichiararne la dimensione.
A questo scopo si utilizza l'apposita istruzione DIM; vediamone un esempio:

```
100 DIM D(4500)
```

comunica al BASIC che l'array numerico D contiene 4500 elementi.
L'istruzione DIM deve precedere qualunque istruzione che faccia uso dell'array e di solito, inoltre, azzera tutti gli elementi dell'array (nel caso di array stringa, riempe di stringhe nulle).
Infine, la DIM non può normalmente essere ripetuta, cioè un array non può essere RIDIMENSIONATO (i BASIC più evoluti consentono di cancellare un array che non serve più, recuperando così spazio in memoria).
In realtà, un array contiene sempre un elemento in più di quanti ne sono dichiarati nella DIM: infatti, l'indice può anche essere zero.
Esiste dunque un ulteriore elemento zero dell'array, il che, di solito, non crea problemi, anzi a volte può essere comodo.
Consideriamo il caso di un array a tre indici:

```
100 DIM A(16,16,16)
```

Apparentemente, questo array contiene 16 x 16 x 16, cioè 4096 elementi. In realtà, poiché esiste l'elemento zero, ne contiene 17 x 17 x 17, cioè ben 4913. Se ogni elemento (che è una variabile numerica) occupa cinque byte, ci sono be 4085 (817 * 5) byte sprecati: può darsi che la memoria non sia più sufficiente! In questo caso, occorre dichiarare:

```
100 DIM A(15,15,15)
```

e fare uso anche dell'elemento zero, a meno che il BASIC non consenta di eleminarlo con l'istruzione OPTION BASE 1, che fa partire tutti gli array dall'elemento uno:

```
100 OPTION BASE 1
110 DIM A(16,16,16)
```

Infine, un'ultima nota. Non è necessario dichiarare gli array che non superino i dieci elementi per indice (a parte l'elemento zero), perciò il programma:

```
10 FOR K = 1 TO 10
20 PRINT "SCRIVI IL NOME, NUMERO: ";K;
30 INPUT NO$(K)
40 NEXT K
```

è corretto e non richiede il DIM: il programma introduce nell'array stringa NO$ i dieci nomi battuti in risposta alle INPUT della linea 30.
La prima volta che il programma incontra NO$(K), esegue automaticamente una DIM NO$(10).

PROGRAMMA STATISTICO

Vediamo di realizzare un programma per effettuare alcuni semplici calcoli statistici su un insieme di dati numerici (somma, valore minimo, valore massimo e valore medio).
Per prima cosa, suddividiamo il lavoro in parti che posssano essere realizzate separatamente:

1) input dei dati;
2) eventuale memorizzazione esterna dei dati raccolti;
3) esecuzione dei calcoli;
4) eventuale memorizzazione esterna dei risultati;
5) stampa dei risultati.

Per ora, non siamo in grado di programmare i punti 2 e 4, ma l'utilità di queste operazioni è evidente: dati e risultati non vanno persi, ma possono essere impiegati in seguito per ulteriori analisi o applicazioni. Per comodità, scriveremo le istruzioni riguardanti il punto 1 a partire dalla linea 100, quelle del punto 2 dalla 200, e così via.
Cominciamo con il punto 1:

```
100 REM --- INPUT DEI DATI ---
110 REM
120 INPUT "QUANTI DATI";ND:REM --- NUMERO DATI
130 DIM D(ND):REM --- SPAZIO PER I DATI
140 FOR I = 1 TO ND
150 PRINT "DATO N. ";I;
160 INPUT D(I):REM --- PRENDE UN DATO
170 NEXT I
180 PRINT "FINE INTRODUZIONE DATI"
```

La linea 120 chiede il numero dei dati che intendiamo introdurre. Questo è necessario per poter indicare al BASIC di quanto spazio avremo bisogno.

La linea 130 (DIM) dichiara, appunto, che useremo un array (vettore) di ND elementi per contenere i dati (ogni elemnto è una normale variabile numerica).
A questo punto occorre effettuare tante istruzioni di INPUT quanti sono i dati da introdurre. La linea 140 apre perciò un ciclo che verrà ripetuto ND volte (ND è il numero dei dati da introdurre).
La linea 150 stampa un prompt (messaggio di avviso) che indica il numero del prossimo dato da introdurre (per comodità). Poiché I è la variabile di controllo del ciclo FOR...NEXT aperto alla linea 140, è anche il numero del dato da introdurre.
La linea 160 è l'input vero e proprio, che dice: metti il dato che viene introdotto da tastiera nell'elemento I dell'array D. Al primo ciclo, I vale 1, perciò è come dire INPUT D(1), cioè il dato entra nel primo elemento dell'array (trascuriamo l'elemento zero).
Al secondo ciclo, I vale 2 ed il dato entra nel secondo elemento, e così di seguito fino al numero stabilito di dati.
Alla fine tutti i dati introdotti sono ben ordinati nel vettore D.
Notiamo che questa parte del programma è tutt'altro che ben fatta: non c'è alcun controllo sui valori introdotti (potrebbero essere troppo grandi o troppo piccoli), nè c'è la possibilità di correggere un valore errato (bisogna reimpostare tutti i dati dall'inizio).
Tuttavia, il vantaggio di aver diviso il programma in piccole parti separate sta proprio in questo: se una va corretta o migliorata, può essere tranquillamente modificata senza toccare le altre.
La seconda parte consiste nel salvare (registrare, memorizzare) i dati su una memoria esterna (dischetto, nastro magnetico).
Per ora scriviamo:

```
200 REM --- SALVA I DATI
```

La terza parte è il calcolo vero e proprio:

```
300 REM --- CALCOLI ---
310 REM
320 SO = 0 : MI = D(1) : MA = D(1)
330 FOR I = 1 TO ND
340 SO = SO + D(I)
350 IF D(I) ‹ MI THEN MI = D(I)
360 IF D(I) › MA THEN MA = D(I)
370 NEXT I
380 VM = SO / ND
```

La linea 320 INIZIALIZZA i valori SO (somma), MI (minimo) e MA (massimo). Precisamente, la somma viene posta a zero, mentre il minimo ed il massimo sono posti uguali al primo dato introdotto (infatti, se non ci sono altri dati il primo è sia il minimo che il massimo).
La linea 330 apre un ciclo che, come al solito, fa passare uno per uno tutti gli elementi dell'array (se si usa I come indice).
La linea 340 aggiunge l'elemento I alla somma (la volta aggiunge l'elemento 1, la seconda l'elemento 2, e così via).
La linea 350 dice: se D(I) (cioè l'elemento I, o meglio, il dato numero I) è minore del minimo, allora questo elemento è il nuovo minimo (e viene assegnato a MI).
La linea 360 fa la stessa cosa per il massimo: se l'elemento è superiore al precedente massimo, subentra a questo.
La linea 370 (NEXT) chiude il ciclo aperto dalla 330.
La 380, infine, calcola il valore medio (somma dei valori divisa per il numero degli elementi).
La quarta parte (memorizzazione dei risultati) è per ora:

```
400 REM --- SALVA I RISULTATI ---
```

Infine, dobbiamo stampare i risultati ottenuti:

```
500 REM --- STAMPA RISULTATI ---
510 REM
520 PRINT "SONO STATI INTRODOTTI ";ND;"
          DATI."
530 PRINT "LA SOMMA DEI VALORI E' ";SO
540 PRINT "IL VALORE MINIMO E' ";MI
550 PRINT "IL VALORE MASSIMO E' ";MA
560 PRINT "IL VALOR MEDIO E' ";VM
570 PRINT "FINE DEL LAVORO."
```

E qui non c'è bisogno di commenti (speriamo!). Con qualche miglioria nell'introduzione dei dati e qualche calcolo statistico in più, questo programmino può venire utile, ad esempio, per l'analisi dei dati raccolti nelle esperienze del laboratorio di fisica.

OPERAZIONI SULLE STRINGHE

Esiste una certa parentela tra STRINGHE e ARRAY: una stringa è una sequenza ordinata di caratteri, così come un array è costituito da una serie di elementi numerati. In alcuni linguaggi, come il PASCAL, le stringhe non sono altro che array (ad un indice, vettori) di caratteri.
Anche alcuni BASIC usano le stringhe in un modo diverso dallo standard. Tuttavia, poiché questo non porta a sostanziali vantaggi, dobbiamo considerarlo un difetto, in quanto i programmi scritti per tali macchine non possono essere facilmente modificati per girare sulla maggioranza degli altri calcolatori. Gli eventuali piccoli vantaggi di impiego non possono compensare questo fondamentale inconveniente.
Al fine di non creare confusione, ci riferiremo alle istruzioni impiegate dalla grande maggioranza dei BASIC per operare con le stringhe, terminando con un accenno ai sistemi non-standard.

BASIC (Parte VII[a])

Due stringhe possono essere CONCATENATE, cioè "sommate" per ottenere una stringa costituita dalle due originarie, poste una di seguito all'altra.
L'operatore di concatenazione delle stringhe è di solito il "+", ma può anche essere un altro carattere, ad esempio "&".
Vediamo un semplice esempio:

```
100 A$ = "Cassa"
110 B$ = "Forte"
120 C$ = A$ + B$:REM CONCATENA STRINGHE
130 PRINT C$

RUN -› CassaForte
```

Un altro possibile esempio è costituito da un programma che simula la funzione SPC$ (che non tutti i calcolatori possiedono), producendo una stringa contenente 200 spazi:

```
100 ST$ = "": REM STRINGA NULLA
110 FOR I = 1 TO 200:REM NUMERO DI SPAZI
120 ST$ = ST$ + " ":REM AGGIUNGE SPAZIO
130 NEXT I
```

La linea 100 pone ST$ uguale alla stringa nulla (cioè ST$ non contiene alcun carattere), poi ad ST$ viene aggiunto (concatenato) uno spazio (linea 120). Il ciclo FOR...NEXT fa ripetere l'aggiunta per il numero di spazi desiderato. Alla fine, la stringa ST$ contiene 200 spazi (se stampata su un video a 40 colonne, cancella cinque righe).
E' possibile "estrarre" parte di una stringa, al limite anche un solo carattere, per mezzo delle funzioni LEFT$, RIGHT$, MID$ (i nomi delle funzioni terminano con "$", in quanto restituiscono delle stringhe).
LEFT$ restituisce una stringa costituita dalla parte sinistra (left) della stringa indicata, per il numero di caratteri specificato.
Vediamo un esempio:

```
100 A$ = "ARANCIA"
110 B$ = LEFT$(A$,4):REM ESTRAE 4 CARATTERI
120 PRINT B$

RUN -› ARAN
```

RIGHT$ fa esattamente la stessa cosa, ma dalla parte destra (right):

```
100 A$ = "ARANCIA"
110 B$ = RIGHT$(A$,3)
120 PRINT B$

RUN -› CIA
```

Da notare che, poiché in BASIC un'espressione può sempre sostituire un valore, potevamo scrivere:

```
100 A$ = "ARANCIA"
110 PRINT RIGHT$(A$,3)

ed era (sintatticamente corretto) anche:

100 PRINT RIGHT$("ARANCIA",3)
```

Ovviamente, si fa uso delle variabili quando occorre impiegare una stringa precedentemente prodotta, oppure conservare la stringa risultante (B$ negli esempi precedenti) per un uso successivo.
La funzione MID$ estrae il numero specificato di caratteri, a partire dalla posizione data:

```
100 A$ = "TASTIERA"
110 PRINT MID$(A$,2,4):REM ESTRAE 4
          CARATTERI A PARTIRE DAL 2

RUN -› ASTI
```

Come si vede, il primo numero specifica il carattere da cui partire ed il secondo il numero di caratteri da estrarre.
Il primo carattere di una stringa è il numero uno.
Scriviamo ora un programma che "rovescia" una stringa, scambiando l'ordine dei caratteri. Possiamo farlo estraendo i caratteri, ad uno ad uno, dal fondo della stringa.
Per farlo, però, dobbiamo sapere quanti caratteri estrarre, cioè la lunghezza della stringa.
La lunghezza di una stringa si può conoscere con la funzione LEN, che restituisce un valore numerico (infatti il nome della funzione non termina con "$") corrispondente al numero dei caratteri contenuti nella stringa (la sua lunghezza, appunto).
In altre parole:

```
PRINT LEN("ADALGISA") -› 8

Ecco il "rovesciatore":

100 INPUT "STRINGA DA ROVESCIARE ";A$
110 B$ = "":REM STRINGA DESTINAZIONE
120 L = LEN(A$):REM NUMERO DI CARATTERI
130 FOR I = L TO 1 STEP -1:REM DAL FONDO
          ALL'INDIETRO
140 B$ = B$ + MID$(A$,I,1):REM NE ESTRAE
          UNO
150 NEXT I
160 PRINT "IL ROVESCO DI ";A$;" E' ";B$
```

La linea 100 chiede la stringa da rovesciare, la 110 si assicura che la nuova stringa B$ non contenga alcun carattere, la 120 calcola la lunghezza della

stringa introdotta da tastiera.
La linea 130 dice di ripetere quanto segue, fino al NEXT della 150, per tante volte quanti sono i caratteri della stringa A$ (calcolati in L), con la variabile I che scende (STEP -1) dal valore massimo L (lunghezza di A$) al valore minimo 1.
Ad ogni ciclo, la linea 140 aggiunge a B$ (concatena) la stringa di lunghezza uno (quindi un solo carattere) estratta da A$ nella posizione indicata da I.
Poiché I parte dal fondo di A$ e scende fino ad uno, i caratteri di A$ vengono estratti a rovescio, ed accumulati in B$.
La linea 160, infine, stampa il risultato.
Come al solito, questo è un programma dimostrativo, che non tiene conto di eventuali errori: non è contemplato, ad esempio, il caso di una stringa nulla introdotta in risposta all'INPUT (cioè un semplice Invio).
Il programma può essere semplificato estraendo i caratteri a partire dal primo ed aggiungendoli in testa (anziché in coda) alla stringa B$.
Siete capaci di farlo?

STRINGHE E NUMERI

E' fondamentale capire la differenza tra stringhe e numeri: solo questi ultimi sono usabili in operazioni aritmetiche, cioè "comprensibili" dal calcolatore.
Le stringhe sono soltanto sequenze di caratteri, che il calcolatore riporta tali e quali.
Perciò:

```
PRINT 2 + 3   -› 5
PRINT 2 + "3"   -› ERROR
PRINT "2" + "3" -› 23 (STRINGHE CONCATE-
                       NATE)
```

E' possibile convertire un numero in una sequenza di caratteri, cioè "stamparlo" dentro una stringa, con la funzione STR$:

```
100 A = 1990
110 Q$ = "ANNO"
120 Z$ = Q$ + STR$(A)
130 PRINT Z$

RUN -› ANNO 1990
```

Z$ contiene dunque la stringa "ANNO 1990", ed il numero 1990 è stato convertito nella sequenza di caratteri "1990".
La funzione inversa, che "legge" un numero da una sequenza di caratteri, si chiama VAL:

```
100 Z$ = "ANNO 1990"
110 A$ = RIGHT$(Z$,4):REM ESTRA ULTIMI 4
120 A = VAL(A$) 130 PRINT A + 1

RUN -› 1990
```

Come si vede, l'istruzione VAL alla linea 120 ha trasformato la stringa "1990" nel numero 1990, che può tranquillamente essere assegnato ad una variabile numerica (A), o impiegato in espressioni (linea 130).

PARTICOLARITA' DELLE STRINGHE

Come abbiamo detto, alcuni BASIC usano le stringhe in modo diverso dalla maggioranza, considerandole array di caratteri.
In questi BASIC, non esistono di solito le funzioni LEFT$, RIGHT$ e MID$.
E' invece possibile copiare o modificare alcuni caratteri di una stringa, semplicemente indicando il primo e l'ultimo carattere interessati.
Per esempio:

```
B$ = A$(3,5)
```

assegna a B$ i caratteri di A$ dal terzo al quinto, esattamente come B$=MID$(A$,3,2).
Più interessante è l'operazione inversa A$(3,5)=B$, che non è possibile eseguire in una sola istruzione nel BASIC standard.
A volte sono impiegate le parentesi quadre invece di quelle tonde.
Di solito è necessario dichiarare la lunghezza delle stringhe, con una DIM simile a quella usata per gli array.
Nel BASIC standard non è necessario specificare la lunghezza massima di una stringa, che può raggiungere di solito i 255 caratteri.
Questo obbliga il BASIC ad un complesso lavoro di HOUSEKEEPING (alla lettera: organizzazione domestica), necessario per fare spazio alle stringhe più lunghe, senza sprecarne per quelle più corte.
Come risultato, ogni programma che usi stringhe finisce prima o poi con il riempire la memoria con resti di vecchie stringhe.
A questo punto il BASIC ripulisce e riordina tutto, compiendo il lavoro di GARBAGE COLLECTION (raccolta della spazzatura) e ricreando spazio in memoria.
Purtroppo, si tratta spesso di un lavoro lento, che può bloccare la macchina anche per vari minuti (il tempo varia con il quadrato del numero di stringhe utilizzate).
Non hanno questo problema i calcolatori che obbligano a dichiarare, all'inizio del programma, la lunghezza massima di ciascuna stringa (in questo caso, ogni stringa sta in certe caselle fisse di memoria).

(continua)

Eccoci giunti al nuovo appuntamento con il nostro corso di programmazione.
In questa parte continueremo a parlare di Programmazione o Notazione Lineare Strutturata, base di qualsiasi linguaggio per computer.

I PROBLEMI DELLA PROGRAMMAZIONE LINEARE STRUTTURATA E L'INDENTAZIONE

Alla quinta lezione di programmazione, disponiamo di un insieme di costrutti di controllo da permetterci di scrivere qualsiasi tipo di algoritmo con semplicità e precisione.
A conferma di ciò, riportiamo di seguito alcuni algoritmi in PLS, derivati direttamente da diagrammi a blocchi e ricostruiti secondo le regole già studiate.

```
ALGORITMO n.1:
dato un numero n intero scrivere se è pari o dispari

begin
leggi n;
if n=(n/2)*2 then scrivi "pari"
else scrivi "dispari"
end
```

ALGORITMO n.2:
dati A e B, scrivere prima il più grande e poi il più piccolo

```
begin
leggi A,B;
if A›B then scrivi "A,B"
else scrivi "B,A"
end
```

ALGORITMO n.3:
dati A,B ›0 calcolare A*B per somme successive

```
begin
leggi A,B;
C=0;
while A‹›0 do begin
C=C+B;
A=A-1
end
scrivi C
end
```

ALGORITMO n.4:
sommare i primi N numeri interi positivi

```
begin
leggi N;
I=1;
S=0;
repeat S=S+I;I=I+1;until I=N;
scrivi S
end
```

Nella scrittura di questi algoritmi in PLS occorre tenere presente alcune regole di fondamentale importanza che i non addetti ai lavori tendono spesso a ignorare.
Mi riferisco in particolar modo al problema della indentazione o allineamento delle righe del programma: se esse cominciano tutte dalla prima colonna, il programma risulta praticamente illeggibile a causa del fatto che, i blocchi di programma, non sono evidenziati per le funzioni che svolgono.
Vi sono fortunatamente due regole generali che possono ammettere eccezioni solo se dettate da un buon senso logico:

1) Se un blocco è interno a un altro, le sue righe devono essere indentate un po' più a destra della colonna rispetto alla quale sono indentate le righe del blocco che lo contiene.

2) Blocchi appartenenti allo stesso livello o sono tutti sulla stessa riga o sono indentati tutti sulla stessa colonna del begin corrispondente.
   Così come la parte ELSE di un costrutto IF-THEN-ELSE deve essere allineata con la parte THEN.

L'inosservanza di queste regole può portare alla scrittura di programmi assolutamente illeggibili ed è perciò una fonte primaria di errori che rendono vano lo strumento della programmazione strutturata.
Il lettore si riferisca ora agli esempi di seguito riportati a conferma di quanto detto.

```
ALGORITMO n.5:
scrivere i quadrati dei numeri da 1 a n sommando i dispari

begin
QUADR=0;
DISP=1;
I=0;
leggi N;
repeat
QUADR=QUADR+DISP;
scrivi QUADR;
DISP=DISP+2;
I=I+1;
until I=N;
end

ALGORITMO n.6:
leggere n numeri e scrivere la loro somma

begin
leggi N;
I=0;
```

```
SOM=0;
repeat leggi NUM;SOM=SOM+NUM;I=I-1 until I=N;
scrivi SOM
end
```

ARRAY O VETTORI, DICHIARAZIONI E COSTANTI SIMBOLICHE

Si consideri il seguente problema: ordinare tre numeri A,B,C letti dall'esterno:

```
begin
leggi A,B,C;
if A›B then
begin AIUTO=A;A=B;B=aiuto end;      *scambia A e B
if B›C then
begin AIUTO=B;B=C;C=AIUTO end;      *scambia B e C
if A›B then
begin AIUTO=A;A=B;B=AIUTO end;      *scambia A e B
scrivi A,B,C;
end
```

I primi due confronti portano avanti il più grande dei tre numeri mentre l'ultimo, se necessario, aggiusterà in seguito i primi due.
Se ora pensiamo però all'ordinamento di N numeri, ci troviamo in un certo imbarazzo.

Con il metodo appena studiato dovremmo associare, a ogni numero, una variabile diversa ed avremmo i seguenti problemi:

1) il programma sarebbe limitato ad un numero N prefissato di numeri da ordinare.

2) La sua lunghezza sarebbe direttamente proporzionale al numero N (per ordinare 1000 numeri occorrerebbe scrivere, come minimo, circa 10000 confronti).

Ovviamente, come vedremo, questi due inconvenienti possono essere eliminati ricorrendo ad un nuovo concetto che trova riscontro in quasi tutti i linguaggi di programmazione.
L'idea fondamentale è quella di associare alla lista a1,a2,...an, di elementi da ordinare, una lista di variabili con indice A[1],A[2],...A[N], nota comunemente come vettore o ARRAY unidimensionale.
In questo modo le variabili hanno un nome collettivo (in questo caso A) e uno individuale, detto indice, denotato da un'espressione che viene valutata prima di ogni operazione di lettura/scrittura sulla variabile.

Per denotare una variabile array, adottiamo la seguente sintassi:

nome variabile [espressione indice]

esempi:

```
VETTORE[N]; A[I]; ARR[K]
```

Si può quindi pensare a un vettore come ad una sequenza ordinata di variabili che vengono individuate a una ad una, dal valore che assume l'indice specificato.
Il vettore A può essere letto dall'esterno con un semplice ciclo FOR:

```
for I=1 to N do leggi A[N]
```

Equivalente alla più scomoda forma:

```
leggi A[1],A[2],A[3],...A[N-1],A[N]!
```

In effetti il ciclo for costituisce il costrutto di controllo più comodo e naturale associato agli array.
Ad esempio, se volessimo sommare ad ogni elemento, la somma di tutti i precedenti potremmo scrivere semplicemente:

```
for I=2 to 100 do A[I]=A[I]=A[I-1]
```

Idealmente ogni array o vettore potrebbe avere un numero di elementi illimitato. E' evidente che le strutture fisiche delle memorie degli attuali elaboratori impongono di stabilire un limite massimo prima di cominciare ad operarvi.

Tutto ciò conduce a due logiche conseguenze immediate:

A) Occorre informare l'esecutore con opportune frasi dichiarative, dette infatti dichiarazioni. Nel nostro caso, la frase dichiarativa corrisponderebbe a:

```
var A: array[1...100] of integer
```

che deve essere inteso dal calcolatore come un comando di riserva per cento elementi per un vettore il cui nome collettivo è A e i cui elementi sono dei numeri interi i cui indici devono essere compresi fra 1 e 100.
Esistono anche numerose applicazioni in cui è utile avere gli indici in un intervallo arbitrario degli interi da -100 a 100.
Quindi la sintassi della corretta dichiarazione degli array è la seguente:

var ‹nome collettivo›:array[‹indice inf...indice sup.›]of‹tipo elementi›

esempio:

A:array [-100..100] of integer

B) Se si tenta di accedere ad elementi inesistenti provoca un errore segnalato dal calcolatore.
Se ad esempio volessimo assegnare il numero 0 all'elemento A[1000] con un vettore da [-100..100] provocheremmo un errore la cui segnalazione sarà: "indici fuori dai limiti".

PARTE DICHIARATIVA E COSTANTI SIMBOLICHE

L'introduzione dei vettori ci costringe ora a fare una netta distinzione fra variabili semplici e variabili con indice.
Per evidenziare questi due tipi di variabili sostanzialmente diverse, conveniamo fin da ora di dichiarare anche le variabili semplici.
Ogni programma sarà così costituito da una parte di dichiarazione a cui farà seguito una parte di esecuzione.

Esempio: calcolo della media di N numeri mediante l'uso di un vettore

```
program MEDIA-ARITMETICA;
var A:array[1..10]of integer;          I parte
I,S:integer;                           I dichiarativa
begin
for I=1 to 10 do S=S+A[I];             I parte
S=0;                                   I esecutiva
for I=1 to 10 do S=S+A[I];             I
scrivi "la media aritmetica è, S/10    I
end
```

Ovviamente questo programma presenta una notevole limitazione: può fare la media solo di 10 elementi.
Un modo di ovviare a questo inconveniente, sarebbe quello di leggere prima la lunghezza N del vettore e poi chiedere all'esecutore di allocare gli elementi all'interno del vettore stesso (allocazione dinamica degli array).
Nella realtà programmativa, questo non è sempre possibile, tuttavia è possibile prendere in considerazione un'altra soluzione che utilizza il concetto di costante simbolica.
Questa soluzione consiste nell'associare un simbolo (o identificatore) ad un particolare valore, ad esempio 10,100,50 etc. con dichiarazioni del genere:

sia ‹nome› = valore, ...
esempio sia N = 100

Il valore tratta poi uno di questi simboli come una qualsiasi variabile.
A differenza delle variabili, le costanti simboliche non possono però subire assegnamenti anche perché, altrimenti, non sarebbero più costanti!

Così, adesso possiamo riscrivere il programma precedente sotto un'altra veste:

```
program MEDIA-ARITMETICA
sia N = 10;                            I parte
var A:array[1..N] of integer;          I dichiarativa
I,S:integer;                           I
begin
for I=1 to N do leggi A[I];            I parte
S=0;                                   I esecutiva
for I=1 to N do S=S+A[I];              I
scrivi "la media è, S/N                I
end
```

In questo modo la costante simbolica N introduce una parametrizzazione del programma.
Sostituendo N=100 al posto di N=10 la variazione viene vista in tutti i punti in cui compare N.

(continua)

DENTRO L'MSX (Parte IV^{a})

Nello scorso numero avevamo concluso parlando del colore nelle diverse modalità di testo e dell'uso della VPOKE per cambiare la tabella dei colori nella Ram, istruzione usata anche per ridefinire il set di caratteri.
In entrambe le modalità testo, la posizione del cursore viene determinata tramite l'istruzione LOCATE X,Y. Nel modo a 40 colonne è possibile visualizzare fino a 40 caratteri su una linea.
Nonostante siano accettati posizioni di colonna fino a 255, solo quelle comprese nell'intervallo 0-39 hanno effetto. Per la modalità a 32 colonne l'intervallo accettabile è da 0 a 31.
Un'alternativa all'istruzione LOCATE 0,0 è la stampa su schermo di CHR$(11).
Il comando LOCATE controlla inoltre la visualizzazione del cursore:

LOCATE 2,4,0 fa si che esso non venga visualizzato
LOCATE 2,4,1 abilita la visualizzazione del cursore

Il comando PRINT, che può essere abbreviato con "?", usa i caratteri standard per la formattazione:

1 - Il punto e virgola implica che il carattere di "ritorno carrello" (CR o Carriage Return, CHR$(13)) non venga stampato dopo l'ultima entità da visualizzare. In questo modo la prossima visualizzazione non partirà dalla prima colonna della linea successiva.

2 - Una virgola implica che la prossima entità visualizzata sia allineata con il prossimo carattere (o zona) di tabulazione presente sulla linea. Ogni zona di tabulazione è ampia 14 colonne.

3 - Il segno "+" serve per concatenare delle stringhe. Per esempio:

A$ = B$ + "XYZ";

TAB(n) e SPC(n) devono seguire l'istruzione di PRINT cioè:

PRINT SPC(4);"NOME

Si noti che il delimitatore finale " può essere assente.
La variante PRINT USING consente di stampare tabelle numeriche o alfanumeriche in un dato formato.
I quattro principali caratteri di controllo sono:

1 - Il punto esclamativo (!): implica che venga stampato solo il primo carattere di una stringa, come PRINT USING "!";"NOME" che darebbe come risultato N

2 - La barra (/): viene usata nella forma "//" per specificare il numero di caratteri della stringa che devono essere stampati. Il numero dei caratteri stampati è infatti due più il numero di spazi tra le due barre, come, ad esempio, PRINT USING "//";"NOME" che darebbe come risultato NOM. Se la lunghezza del campo supera quella della stringa, vengono aggiunti degli spazi.

3 - Il segno di E commerciale (&): viene usato per inserire una seconda stringa in una prima a partire dalla posizione in cui è posta la &; quindi

Q$="NOME":? USING "IN &";Q$ darebbe come risultato IN NOME.

Si noti che nel caso venissero fornite più sottostringhe, la sequenza mostrata in precedenza verrebbe ripetuta per ciascuna di esse.
Per esempio:

Q$="A":Z$="B":? USING "SU $";Q$;Z$ darebbe come risultato SU ASU B.

4 - Il segno di diesis (#): consente di stampare i numeri fissando il numero di cifre prima e dopo la virgola decimale. Se il valore è più corto, vengono aggiunti degli 0. Per esempio:

? USING "####.####";2,4.684 darebbe 2.004.68.

Se viceversa il campo è troppo corto per contenere il dato, allora viene stampato il segno % davanti al valore, oppure viene arrotondato. Per esempio:

? USING"##";22444 darebbe come risultato %22444

mentre

? USING"##.##";22.1234 darebbe 22.12.

Se il campo è troppo grande il valore viene giustificato a destra o vengono aggiunti degli zeri.
Per esempio:

? USING"##.##";22.2 darebbe come risultato 22.20

mentre con

? USING "#####";22 si otterrebbe 22.

Il numero non può essere più lungo di 24 cifre, nel qual caso verrebbe segnalata una situazione di errore.
Altri caratteri di controllo per i numeri, usati insieme al segno "#", sono: **, $$, la virgola, +, - e ^^^^ (detto "carat").
Un segno più (+) alla destra o alla sinistra del simbolo diesis (#) implica che venga stampato il segno del valore numerico, rispettivamente alla destra o alla sinistra del valore stesso.
In modo analogo, un segno negativo dopo l'ultimo di

una serie di "#" provoca la stampa del segno meno dopo il valore, se questo è negativo.
Quindi:

```
? USING "##-";-2 stamperà 2-
```

La combinazione doppio asterisco (**) viene usata sulla sinistra dei simboli # e fa sì che gli spazi iniziali, che risultano da una giustificazione a destra di un valore, vengano riempiti con asterischi; quindi:

```
? USING "**#.#";4.2 dà come risultato **4.2

? USING "**##.#";4.2 dà invece ***4.2
```

Il segno di doppia lira (che qui indichiamo con $$) provoca la stampa del simbolo di lira prima del valore numerico in questione. Questo carattere di controllo non può essere usato in coincidenza del "carat" (^^^^).
Se si pone una virgola alla sinistra del punto decimale, il numero viene stampato con una virgola a sinistra di ogni gruppo di tre cifre, vale a dire:

```
? USING "#####,.#";2222.2 provoca la stampa di 2,2222.2
```

Infine il quadruplo carattere (^) viene usato per stampare un valore numerico in forma esponenziale; come esempio si consideri:

```
? USING :##.#^^^^";22.4 implica la stampa di 2.2E+01
```

Lo schermo può essere pulito in tutte le modalità di visualizzazione mediante il comando CLS. In entrambe le modalità testo, un'alternativa è la stampa di CHR$(12). La coordinata X da POS(x) (dove X è un argomento fittizio, che può quindi essere sostituito da un qualsiasi valore).

MANIPOLAZIONE DELLA RAM VIDEO

Per ottenere che i caratteri siano visualizzati con colori diversi da quelli impostati globalmente per mezzo del comando COLOR nella modalità testo a 32 colonne, si deve andare a modificare il byte appropriato nella tabella dei colori della VRAM, scegliendo una nuova combinazione primo-piano/sfondo.
Come già accennato, ciascuno dei 32 byte della tabella dei colori determina i colori di un insieme di 8 dei 256 caratteri disponibili. Come conseguenza di ciò, per visualizzare del testo in colori diversi da quelli di default è necessario copiare parte dell'insieme dei caratteri in un'altra sezione della tabella di generazione dei modelli.
Ecco come è allocata la memoria video nella modalità testo a 32 colonne:

```
TABELLA GENERATRICE DEI MODELLI: 0 - 2047
TABELLA DEI COLORI: 8192 - 8223
TABELLA DEGLI ATTRIBUTI DEGLI SPRITE: 6912 - 7039
TABELLA DEI NOMI: 6144 - 6911
TABELLA DEI MODELLI DEGLI SPRITE: 14336 - 16383
```

Si noti che i caratteri grafici occupano i primi 2 blocchi di definizione nella tabella dei modelli. Questo perché i primi 32 codici carattere non sono stampabili. I caratteri grafici possono quindi essere visualizzati sullo schermo solo stampando (PRINT) CHR$(1) + CHR$(65).
Un altro carattere non standard ha codice 255: provoca la visualizzazione inversa del carattere su cui è posizionato il cursore.
Viene aggiornato circa ogni 1/50esimo di secondo.

ESEMPIO DI TESTO MULTICOLORE

Per ottenere che un testo venga visualizzato in lettere maiuscole con primo piano giallo e sfondo nero, sono necessari i seguenti passi:
Copiare le 26 definizioni dei caratteri in un'altra sezione della tabella di generazione dei modelli. L'insieme di caratteri viene definito usando 8 byte per ciascun carattere. L'insieme di caratteri viene definito usando 8 byte per ciascun carattere.
Nella modalità testo a 32 colonne la tabella dei modelli si estende, nella VRAM, dalla locazione 0 fino alla 2048.
Poiché ogni carattere necessita di otto byte di definizione, il primo byte che deve essere copiato si trova alla locazione:

```
ASC("A") * 8 = 520

e il byte finale della definizione dei caratteri sarà

ASC("Z")*8+7=727
```

I 'nuovi' caratteri maiuscoli sovrascriveranno parte dell'insieme di caratteri esistenti. Sostituiremol i caratteri 145-170, che sono definiti da 145*8=1160 a 170*8+7-1467, come segue:

```
10 SCREEN 1:FOR X=0 TO 207 : VPOKE 1160 + X , VPEEK (520 + X) : NEXT X
```

I quattro byte della tabella dei colori che determinano i colori che assumeranno i 32 caratteri dal 144 al 176 risiedono alle locazioni 8210-3.
Il valore contenuto dai quattro bit più significativi in ciascuno di questi determina il colore del primo piano, mentre quelli meno significativi influiscono sul colore dello sfondo. Quindi, con

```
VALORE = GIALLO * 16 + NERO = 10 * 16 + 1 = 161
```

DENTRO L'MSX (Parte IV^{a})

la prossima linea di programma è:

20 FOR X=0 TO 3:VPOKE X+8210,161:NEXT X

Se ora si prova a far visualizzare il CHR$(145) apparirà una 'A' gialla in capo nero.
Si noti che se si usa il comando COLOR tutti i 32 byte della tabella del colore sarebbero impostati con i colori prescelti.
Per ottenere nuovamente dei colori alternativi sarebbe necessario ripetere la linea 20.
I caratteri sono giustificati a sinistra: quindi quando vengono visualizzati su uno sfondo che contrasta con il colore dello schermo, può succedere che la parte sinistra del carattere appaia confusa.
Questo metodo per determinare i colori dello schermo, sebbene macchinoso per certi aspetti, permette di realizzare delle schermate efficaci dal punto di vista grafico in modo piuttosto semplice.
Per esempio, una schermata che consista principalmente di 4-8 caratteri può essere 'animata' o resa lampeggiante manipolando solo due byte della tabella del colore.
Il programma che viene presentato di seguito consente di ridefinire interamente l'insieme dei caratteri, di salvarlo e caricarlo da nastro.
I tasti di controllo del cursore sono usati per muoversi sulla matrice dei caratteri e la barra di spaziatura viene utilizzata per invertire il valore del particolare bit su cui si è posizionati.
Eccovi, per concludere, un'esempio di programma per la definizione dell'insieme di caratteri:

```
100 'DEFINIZIONE CARATTERI'
110 FOR X=0 TO 12:READ A$,B$:POKE 380000+X,VAL("&H"+A$):POKE 38200+X,VAL("&H"+B$):NEXT X
120 DEFUSR=38000:DEFUSR2=38200
130 DATA 21,21,00,40,00,9C,11,11,40,00,9C,00,01,00,00,08,08,CD,CD,59,5C,00,00,C9,C9
140 ON KEY GOSUB 240,270,290,430:KEY(1) ON:KEY(2) ON:KEY(3) ON:KEY(4) ON
150 ON STRIG GOSUB 400:STRIG(0) ON
160 KEY OFF:COLOR 14,1,1:SCREEN 1,0:CLS
170 LOCATE 2,1:PRINT "F1..Save F3..Change CHR$:LOCATE 2,3:PRINT "F2..LOAD F4..COPY CHR$"
180 FOR X=0 TO 7:VPOKE 14336+X,VPEEK(224+X):NEXT X:PUT SPRITE 0,(144,47),8,0
190 FOR X=373 TO 375:VPOKE X,0:NEXT X:VPOKE 371,24:VPOKE 372,24:
200 A$=STRING$(8,46):FOR X=0 TO 7:LOCATE 16,X+6:PRINT A$:NEXT X
210 GOSUB 290:X1=1:Y1=1
220 X2=STICK(0):IF X2=0 THEN 220
230 ON X2 GOTO 350,220,360,220,370,220,380
240 Z=UST(2):LOCATE 2,20:PRINT "REGISTRA E POI RITORNA"
250 A$=INPUT$(1):IF ASC(A$)‹›13 THEN 250 ELSE BSAVE "CHARS",40000,42047
260 LOCATE 2,20:PRINT STRING$(28,32):RETURN
270 LOCATE 2,20:PRINT "LOADING...":BLOAD"CHARS":Z=USR2(2)
280 LOCATE 2,20:PRINT STRING$(28,32):RETURN
290 LOCATE 0,20:PRINT "DIGITA N. DI CHR$‹Invio›":GOSUB 470:CN=XX
300 FOR X=0 TO 7:A$(X)=BIN$(VPEEK(CN*8+X)):NEXT X 310 FOR X=0 TO 7:IF LEN(A$(X))‹8 THEN A$(X)=STRING$(8-LEN(A$(X)),79)+A$(X)
320 FOR BT=1 TO 8:LOCATE 15+BT,X+6:IF MID$(A$(X),BT,1)="1" THEN PRINT CHR$(219) ELSE PRINT CHR$(46)
330 NEXT:NEXT:LOCATE 2,11:PRINT "CARATTERE ";CN:LOCATE 7,13:PRINT CHR$(CN)
340 RETURN
350 IF Y1=1 THEN 220 ELSE Y1=Y1-1:GOTO 390
360 IF X1=8 THEN 220 ELSE X1=X1+1:GOTO 390
370 IF Y1=8 THEN 220 ELSE Y1=Y1+1:GOTO 390
380 IF X1=1 THEN 220 ELSE X1=X1-1
390 PUT SPRITE 0,(136+X1*8,39+Y1*8),8,0:FOR X=1 TO 80:NEXT:GOTO 220
400 LOCATE 15+x1,5+Y1:IF MID$(A$(Y1-1),X1,1)="1" THEN MID$(A$(Y1-1),X1,1)="0":PRINT CHR$(46) ELSE MID$(A$(Y1-1,X1,1)="1":PRINT CHR$(219)
410 NN=CN*8+Y1-1:IF MID$(A$(Y1-1),X1,1)="1" THEN VPOKE NN,VPEEK(NN)ORE(2^(8-X1)) ELSE VPOKE NN,VPEEK(NN)AND(255-(2^(8-X1)))
420 RETURN
430 LOCATE 0,20:PRINT "CHR$ DA COPIARE‹Invio›":GOSUB 470:C1=XX
440 LOCATE 0,20IORINT "CHR$ DA RIMPAZZARE":GOSUB 470:C2=XX
460 LOCATE 0,20:PRINT STRING$(28,32):RETURN
470 C1$=""
480 X$=INKEY$:IF X$="" THEN 480
490 IF ASC(X$)‹32 AND ASC(X$)›27 THEN 480
500 IF ASC(X$)‹›13 THEN C1$=C1$+X$:GOTO 480 ELSE XX=VAL(C1$)
510 IF XX›255 OR XX‹0 THEN 470
520 LOCATE 0,20:PRINT STRING$(28,32):RETURN
```

TENDENZE

E' terminato un altro anno, il primo di un nuovo decennio, e con la fine di un anno e l'inizio di quello nuovo si fa sempre un bilancio di ciò che è stato. Dal bilancio dell'ultimo anno e dell'ultimo decennio possiamo cercare di capire ciò che sarà, dove andrà il mercato e l'intero mondo dell'informatica. Vediamo quali sono le tendeze del mercato per questo nuovo anno, per il prossimo decennio.

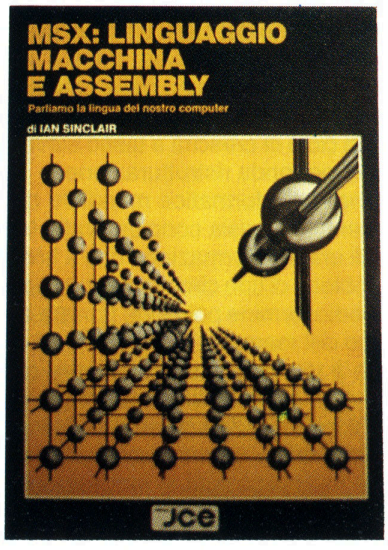
TELECOMUNICAZIONI

Le telecomunicazioni contribuiscono per oltre metà del business complessivo dell'information technology ed il settore tecnologico in maggiore sviluppo e che sta conoscendo le più profonde trasformazioni.
Il 1989 ha visto finalmente la concretizzazione di un dibattito che aveva gettato le sue basi già vent'anni fa. Stiamo parlando della Rete Numerica Integrata "ISDN".
A partire dal 1988 infatti hanno avuto il via le prime realizzazioni concrete e i relativi apparati stanno diventando finalmente disponibili sul mercato.
Nel corso dell'anno passato l'introduzione delle reti ISDN nei paesi Cee hanno proceduto in modo abbastanza spedito, ed è stato completato circa il 60/70% dei lavori di standardizzazione. A fine anno il servizio commerciale ISDN era offerto in quattro paesi europei e in altri quattro erano disponibili servizi su basi sperimentali. Tutti gli altri paesi Cee prevedono l'introduzione del nuovo sistema entro il 1992.
Anche le tecnologie di trasmissione hanno riscontrato un evoluzione degna di nota. Nelle moderne reti di trasmissione si assiste ormai a una estensione del ruolo dei sistemi trasmissivi dal puro trasporto ad altre aree in forte sviluppo come la radiotelefonia mobile, le reti di distribuzione e la gestione degli istradamenti. E in questo senso le aziende che operano nel settore, vista questa diversificazione, hanno maggiori possibilità di ampliare le proprie prospettive.
A fronte di questo aspetto l'introduzione delle fibre ottiche sta creando la condizione per assicurare la connettività globale e la capacità tecnologica di supportare applicazioni integrate.
Ma il 1990 è stato sicuramente l'anno della telefonia mobile. La diffuzione delle reti cellulari ha favorito lo sviluppo di una serie di servizi a partire dall'ormai tradizionale telefono in macchina, ai Telepoint, ai telefoni installati sugli aerei, ma sopprattutto alla telefonia personale, il Personal Communications Networks.
Nel nostro paese, che proprio sulla telefonia personale sta attraversando un momento di vera "corsa al telefono portatile", la scelta tecnologica è caduta su un sistema-ponte analogico che opera sulla frequenza dei 900 Mhz.
Un ultimo sguardo a un altro fenomeno tecnologico, ma ormai anche di costrunme, che nel 1990 ha compiuto un vero e proprio rilancio. Stiamo parlando naturalmente del Videotel, che iniziati i suoi primi passi nel lontano 1986, e dopo una fase di sperimentazione contrassegnata da una serie di difficoltà, sta finalmente decollando.
Il concessionario pubblico ha seguito la felice esperienza del Minitel francese e ha avviato una rivoluzione tecnologica e di marketing che ha consentito da una parte una vertiginosa crescita quantitativa e qualitativa dei servizi e dei fornitori di informazioni (in Francia sono chiamati "Serveurs") e dall'altra un'impennata dei possessori dei terminali, visto che vengono noleggiati a un canone mensile di L. 7000. E a fronte di questa "rivoluzione" i risultati non hanno tardato ad arrivare: alla fine del 1989 gli abbonati erano pressochè raddoppiati, raggiungendo la ragguardevole cifra di 77.500. E gli abbonati sono in continua crescita, mese dopo mese.

IL SETTORE INFORMATICO

Il biennio 1989-90 è stato protagonista di una sequenza di eventi che consentono di prevedere con sufficiente approssimazione le tendenze dei singoli mercati nel corso dell'immediato futuro, e questo a dispetto delle notizie piuttosto negative che hanno scosso il mercato mondiale soprattutto nel comparto dell'informatica.
Si è molto parlato, tra la fine del 1989 e l'inizio del 1990 della crisi dell'industria informatica mondiale, che si è esplicitata in una forte riduzione di utili per alcune delle maggiori aziende americane ed europee, tanto da spingere molti osservatori a dichiarare chiuso un ciclo di crescita a tasso elevato e ad annunciare una fase di profonda ristrutturazione.
In effetti l'evoluzione dell'informatica mondiale non sembra attraversare una semplice crisis congiunturale: le tendenze in atto sono frutto di una trasformazione che a breve termine sta producendo problemi di tipo circoscritto e apparentemente negativi, ma a medio termine potrebbe portare a nuove opportunità di sviluppo del mercato.
A questo proposito è interessante sottolineare che uno dei problemi principali che ha determinato la congiuntura sfavorevole e senz'altro il forte disallineamento tra l'elevata velocità di crescita della domanda, determinata dall'incapacità dell'utenza di tradurre in applicazioni le innovazioni tecnologiche lanciate sul mercato dai costruttori.
Ed è proprio su questo punto che oggi si registrano i maggiori risultati sfavorevoli, ma che in futuro si potrà ricostruire la capacità di fare profitto dei produttori di tecnologia informatica: se questi ultimi infatti sapranno fornire agli utenti gli strumenti culturali per sfruttare la tecnologia di volta in volta disponibile, la capacità di assorbimento del mercato risulterà aumentata, e il mercato si ristabilizzerà sui livelli di crescita a cui ci si è abituati per un intero decennio. Il personal computer tuttavia non ha risentito che in minima parte della congiuntura, e soprattutto in Italia i dati di vendita sono rimasti su livelli soddisfacenti. Il segmento di mercato che risulta ancora trainante è quello dei pc di classe 286, cioè basati su microprocessore Intel 80286, che hanno un parco installato di oltre 800mila unità, quasi il 35% del totale.
I sistemi di fascia bassa, basati su microprocessore 8086, che secondo le previsioni degli anni passati avrebbero dovuto registrare un declino, si sono mantenuti su buoni livelli di vendita e diffusione, e questo trend verrà conservato ancora per almeno due anni. Il parco installato in Italia è ormai di 813mila macchine, quasi il 50% del totale.
I personal computer basati su microprocessore 80386sx, presentati alla fine del 1988, hanno conseguito un notevole successo: 54 mila unità vendute, grazie al rapporto fra prezzo e potenza di calcolo che li ha suggeriti come soluzione ottimale per utenti di sistemi della fascia inferiore, ma anche soluzione meno costosa per utenti che in precedenza avrebbero optato per sistemi 80386.
Questi ultimi hanno confermato nel corso del 1990 il successo atteso, ponendosi in aperta concorrenza con i microcomputer di fascia bassa e le workstation tecnico-scientifiche grazie alla disponibilità di schede acceleratrici, monitor ad alta risoluzione, hardware e software di rete locale e memorie di massa di sempre maggiori dimensioni.
Un discorso a parte merita infine il segmento dei personal computer portatili, che ha imboccato la strada del successo duraturo e progressivo grazie alla loro capacità di racchiudere in un ingombro limitato le caratteristiche e la potenza di calcolo dei sistemi da tavolo. Mentre in passato i costruttori presentavano in chiave di marketing il plus della trasportabilità e dell'autonomia dalla rete elettrica, oggi l'approccio è mutato a favore della compattezza e del minimo ingombro sulla scrivania, cercando con questo di catturare l'attenzione di professionisti, imprenditori e dirigenti che cercano nel personal computer qualcosa in più rispetto al pc da scrivania.

IL LIBRO DEL MESE



MSX: LINGUAGGIO MACCHINA E ASSEMBLY
di Ian SINCLAIR
Edito da: JCE
Pagine: 212
Prezzo: 25.000 lire

In genere, chi usa un computer MSX e sa già scrivere ed utilizzare programmi BASIC, è spesso riluttante ad intraprendere il passo successivo, cioè imparare a servirsi del linguaggio usato dal microprocessore Z80, il cuore dello standard MSX.

Questa rilutttanza è comprensibile. Molti libri che trattano dello Z80 presumono che il lettore conosca già non solo i termini usati, ma anche gran parte del funzionamento del chip del microprocessore.

Altri libri, invece, appaiono agli occhi del principiante, come se fossero scritti in una lingua sconosciuta senza sottotitoli.

Altri ancora hanno un inizio promettente, ma poi perdono per strada il lettore insesperto, perchè affrontano improvvisamente temi assai più difficili o argomenti, come le routines aritmetiche, di ben scarsa utilità per la maggior parte dei lettori.

Questo libro è diretto al vero principiante del linguaggio macchina: chi possiede un computer MSX e sa programmare in BASIC, ma non ha la più pallida idea di quanto avvenga all'interno del suo computer.

L'autore di questo libro non si è proposto lo scopo di far diventari i lettori degli esperti nella programmazione in linguaggio macchina dello Z80, perchè potrebbero arrivarci solo con molta esperienza, molta pratica, molte letture ed un reale desiderio di risolvere tutti i problemi.

L'autore non pretende neppure di spiegare tutto ciò che lo Z80 è in grado di fare.

Ciò che crede di poter ottenere, invece, è di presentare ai lettori l'inizio di un grosso argomento e di fare capire alcuni dei perchè e dei percome della programmazione dello Z80.

Un nuovo, immenso mondo si apre davanti agli occhi dei lettori, che potranno capire meglio come funziona il computer e che potranno finalmente comprendere libri più complessi sulla programmazione in linguaggio macchina.

Bisogna però chiarire bene un punto: questo tipo di programmazione non è mai facile.

Può diventare familiare, può diventare un normale modo di operare, ma non sarà mai facile.

Anche l'imparare è un compito che richiede impegno, un certo sforzo per capire che cosa avviene ed un po' di tempo per fare esercizio sul vostro computer.

In conclusione, questo è davvero un buon libro e per aiutare il lettore, l'autore ha incluso la descrizione di un utilissimo programma: lo ZIN assembler, già pubblicato negli scorsi numeri di questa rivista.

GRUPPO EDITORIALE
INTERNATIONAL EDUCATION

Caro amico,
conosciamo il tuo interesse a quanto, di seguito, siamo in grado di offrirti in forma esclusiva ed inedita.
Abbiamo selezionato la più completa ed interessante raccolta di programmi gioco e utility per l'utilizzo del tuo Msx che possono soddisfare ogni particolare esigenza.
Questi ti verranno pubblicati con sequenza mensile, in elenchi da 100 pezzi, nei quali potrai scegliere e richiedere quelli che ti interessano particolarmente: otterrai così uno o più dischi che conterranno un minimo di 5 programmi ai seguenti prezzi: L. 7.000 per ogni gioco Msx 1, L. 9.000 per ogni gioco Msx 2, L. 10.000 per ogni utility, L. 2.500 per ogni dischetto usato per contenere i 5 programmi da te scelti. La spedizione sarà in contrassegno, cioé pagherai al postino quando riceverai i dischi. Il costo della spedizione sarà aggiunto a quello totale dei programmi da te scelti.
Ti saluto cordialmente e attendo tue notizie.

GRUPPO EDITORIALE INTERNATIONAL EDUCATION s.r.l.
Viale Famagosta, 75 - 20142 Milano - Tel. 02 - 89502288 r.a. - Telefax: 02 - 8466834 - Telex: 352191 GRELIN I
Cod. Fisc. e P. IVA 08428290152 - Cap. Soc. £. 99.000.000 - C.C.I.A.A. di Milano 1223389 - Reg. Trib. Milano 261739/6816/39

CODICE	NOME PROGRAMMA	AUTORE	TIPO	Kb	MSX
179	10 YARD FIGHTER	IREM	FOOT.AMER.	64	1
361	180	MASTERTRON	FRECCETTE	64	1
589	3D BOMBERMAN	HUDSON	ARCADE	32	1
79	3D GOLF	T&E	SPORT	32	1
90	3D SUPER BOWL	ALLIGATA	SPORT	32	1
1171	737 FLIGHT SIMULATOR	MIRRORSOFT	SIMULATORE	32	1
13102	ABADIA DEL CRIMEN	OPERASOFT	ARCAED/AVV	64	2
1545	ACE OF ACE		SIM/AEREO	64	1
1778	ADAM & EVE		ARCADE	64	2
1997	ADONIS		ARCADE	32	1
21117	AFTER BURNER	SEGA	ARCADE	64	1
2397	AGUA BRAVA		SPORT	32	1
25113	ALCAZAR	ACTIVISION	ARCADE/AVV	32	1
2750	ALIBABA	ICM	ARCADE	64	1
2996	ALIENS		ARCADE/AVV	64	2
31101	ALIENS 2	OCEAN	ARCADE/AVV	64	1
3347	ALPHA BLASTER	AACKOSOFT	SPAZIALE	64	1
3531	ALPHAROID		ARCADE	64	1
3797	AMAROUTE	MASTERTRON	ARCADE	64	1
3927	AMIDA	JVC	ARCADE	32	1
4189	ANGLEBALL	MASTERTRON	ARCADE	64	1
4335	ANIMALS WARS	ASCII	ARCADE	64	1
452	ANTARTIC ADVENTURE	KONAMI	ARCADE	64	1
4723	APEMAN STRIKES BACK	AACKOSOFT	ARCADE	64	1
49188	ARCHIVIAZ. ADVANCED	COMP.HOUSE	DATABASE	32	1
5120	ARKANOID	TAITO	ARCADE	64	1
5384	ARKOS 2	ARCADIA	ARCADE/AVV	64	1
5528	ARMY MOVES 1	DINAMIC	ARCADE	64	1
5762	ARQUIMEDES XXI	DINAMIC	ADV/ARCADE	64	1
5961	ASSO	COLFAX	SPAZIALE	32	1
614	ATLETIC LAND	KONAMI	ARCADE	64	1
6326	AUFWIEDERSEN MONTY	GREMLIN	ARCADE/AVV	64	1
6535	BACK TO THE FUTURE	UNIVERSAL	ARCADE	64	1
6735	BALANCE	HAL	ARCADE/EDU	32	1
693	BANANA		ARCADE	64	1
7141	BANK PANIC	SEGA	ARCADE	64	1
73113	BARBARIAN	MASTERTRON	ARCADE/AVV	64	1
753	BASEBALL	KONAMI	SPORT	64	1
7782	BASEBALL MSX	MATSUSHITA	SPORT	32	1
79192	BASIC DISK 1	VARIE	PROG.BASIC	32	1
81194	BASIC DISK 3	VARIE	PROG.BASIC	32	1
83166	BASKET MASTER	DINAMIC	SPORTIVO	64	1
8519	BATMAN	OCEAN	ARCADE/AVV	64	1
8773	BATTLE CHOPPER	METHODIC S	ARCADE	64	1
8942	BATTLE FOR MIDWAY	PSS	SIM/GUERRA	64	1
9128	BC'S QUEST FOR TIRES	TOSHIBA	ARCADE	64	1
9330	BEAMRIDER	ACTIVISION	ARCADE	32	1
954	BEE & FLOWERS	MILAN SOFT	ARCADE	32	1
9771	BIRDIE	ASCII	ARCADE	32	1
9874	BLAGGER	ALLIGATA	ARCADE	32	1

CODICE	NOME PROGRAMMA	AUTORE	TIPO	Kb	MSX
234	10TH FRAME	OCEAN	BOWLING 3D	64	1
4161	1942	SEGA	SHOOTEMUP	64	2
667	3D FIRE		SPAZIALE	64	2
84	3D MAZE	MSX CM	LABIRINTO	32	1
1014	3D WATER DRIVER	COLFAX	ARCADE	32	1
1237	A VIEW TO A KILL	DOMARK	ARCADE/AVV	64	1
14164	ABU SIMBEL PROFANATI	DINAMIC	ARCADE/AVV	64	1
1615	ACTMAN	ASCII	ARCADE	32	1
1669	ADDICTA BALL	ALLIGATA	ARCADE	64	1
20169	ADRIAN MOLE DIARYS	LEVEL 9	AVVENTURA	32	1
22100	AFTEROIDS	ZIGURAT	ARCADE	64	1
2489	AGUIA DE FOGO		ARCADE	32	1
2682	ALE HOP!	TOPO SOFT	ARCADE	64	1
28166	ALIEN 8	ULTIMATE	ARCADE/AVV	32	1
3089	ALIENS	OCEAN	ARCADE	64	1
32169	ALIENS - THE MOVIE	ACTIVISION	ARCADE	64	1
342	ALPHA SQUADRON	AG	ARCADE	64	1
3682	ALPINE SKI	METHODIC	SPORT	32	1
3828	AMERICAN TRUCK	TELENET	ARCADE	64	1
4020	ANGELO	ASCII	ARCADE	32	1
4262	ANIBASKET	LEBEN PRO	SPR/ARCADE	64	1
4441	ANTARES	JULIET	SPAZIALE	64	1
460	ANTY	MOORWOOD	ARCADE	32	1
48103	ARAMO	SEIN	ARCADE	32	1
500	ARCHIVIO		ARCHIVIO	32	1
5264	ARKOS 1	ARCADIA	ARCADE/AVV	64	1
5484	ARKOS 3	ARCADIA	ARCADE/AVV	64	1
5628	ARMY MOVES 2	DINAMIC	ARCADE	64	1
58112	ASPAR G.P. MASTER	DINAMIC	SPORTIVO	64	1
6045	ASTRO PLUMBER	BLUERIBBON	ARCADE	32	1
6217	ATTACK KILLER TOMATO	GLOBAL	ARCADE/AVV	64	1
6421	AVENGER	GREMLINARC	ARCADE/AVV	64	1
6622	BACKGAMMON		SCACCHIERA	64	1
6880	BALL BLAZER	LUCASFILM	ARCADE	64	2
7089	BANDAI	TSUBUBAYA	ARCADE	32	1
7276	BANZAI		ARCADE	32	1
7442	BARN STORMER	ELECTRIC	ARCADE	64	1
7626	BASEBALL CRAZE	HAL	SPORT	64	1
785	BASIC COMPILER	MICROSOFT	COMPILER	64	1
80193	BASIC DISK 2	VARIE	PROG.BASIC	32	1
8234	BASKET	DINAMIC	SPORT	64	1
84140	BASTARD	XEIN	ARCADE/AVV	64	2
86188	BATTERIA		MUSICA	32	1
8842	BATTLE CROSS	OEC	SPAZIALE	32	1
90141	BBS DISK	VARIE	FILES BBS	64	1
9231	BEACH HEAD	ACCESS	ARCADE	64	1
9417	BECKY	MIA	ARCADE	32	1
9671	BINARY LAND	HUDSON	ARCADE	32	1
9898	BLACK BEARD	TOPO SOFT	ARCADE/AVV	64	1
10011	BLASTEROIDE	GRANDSLAM	SPAZIALE	64	1

TIENI QUESTO ELENCO

INSIEME AI SUCCESSIVI CONTRIBUIRA' A FORMARE UNA COMPLETA ED INTERESSANTE RACCOLTA DI PROGRAMMI GIOCO E UTILITY CHE POTRANNO SODDISFARE OGNI TUA ESIGENZA

DESIDERO RICEVERE I SEGUENTI PROGRAMMI (MINIMO 5) DI CUI TRASCRIVO N° CODICE:

Tenendo conto che ogni gioco per MSX 1 costa £ 7.000, per MSX 2 £ 9.000, per ogni UTILITY £ 10.000, per ogni dischetto usato per contenere i programmi £ 2.500 cad., per cui dovrò riconoscervi la spesa totale di £ ..... più spese postali

NOME ..... COGNOME ..... VIA .....

..... C.A.P. ..... CITTÀ ..... PROV. (.....)

TEL. .....

Inviare in busta chiusa a:

GRUPPO EDITORIALE INTERNATIONAL EDUCATION S.r.l.
Viale Famagosta 75 - 20124 Milano - Tel. 02 / 89502288 r.a. - Telefax 02 / 8466834

MUSIC FOR LOVERS

- ENDLESS NIGHT CD 01034
- BEAUTIFUL GIRLS CD 01035
- SUNLIGHT MELODIES DGC 1020

3 COMPACT DISC
AL PREZZO DI L. 29.900

Scopri nuove magiche atmosfere con le più belle melodie d'amore

SERIE "STRUMENTALI"

Desidero ricevere l'offerta "MUSIC FOR LOVERS 2" **cod. CD9**
Allego assegno ☐ ricevuta versamento ☐
+ L. 2.500 quale contributo spese postali

NOME COGNOME

VIA N.

C.A.P: CITTÀ

Firma

Compilare il coupon allegando ricevuta (o fotocopia) del versamento effettuato sul C/C n. 11319209 intestato a **Gruppo Editoriale International Education srl** oppure assegno non trasferibile e spedire a:

Gruppo Editoriale International Education srl
viale Famagosta 75
20142 Milano

SPLENDIDI INEDITI DEI MOSTRI SACRI DEL JAZZ

7 COMPACT DISC AL PREZZO DI L. 84.000

- ○ CHARLIE PARKER CDJJ 610
- ○ BENNY GOODMAN CDJJ 609
- ○ COUNT BASIE CDJJ 604
- ○ SIDNEY BECHET CDJJ 603
- ○ DIZZY GILLESPIE CDJJ 606
- ○ DUKE ELLINGTON CDJJ 602
- ○ LIONEL HAMPTON CDJJ 605

Desidero ricevere l'offerta "JAZZ" **codice CD7**
Allego assegno ☐ ricevuta versamento ☐
+ L. 2.500 quale contributo spese postali

NOME COGNOME

VIA N.

C.A.P. CITTÀ

Firma

Compilare il coupon allegando ricevuta (o fotocopia) del versamento effettuato sul C/C n. 11319209 intestato a **Gruppo Editoriale International Education srl** oppure assegno non trasferibile e spedire a:

Gruppo Editoriale International Education srl
viale Famagosta 75
20142 Milano

JAZZ