NOVEMBRE 1990
ANNO III

# MSX DISK

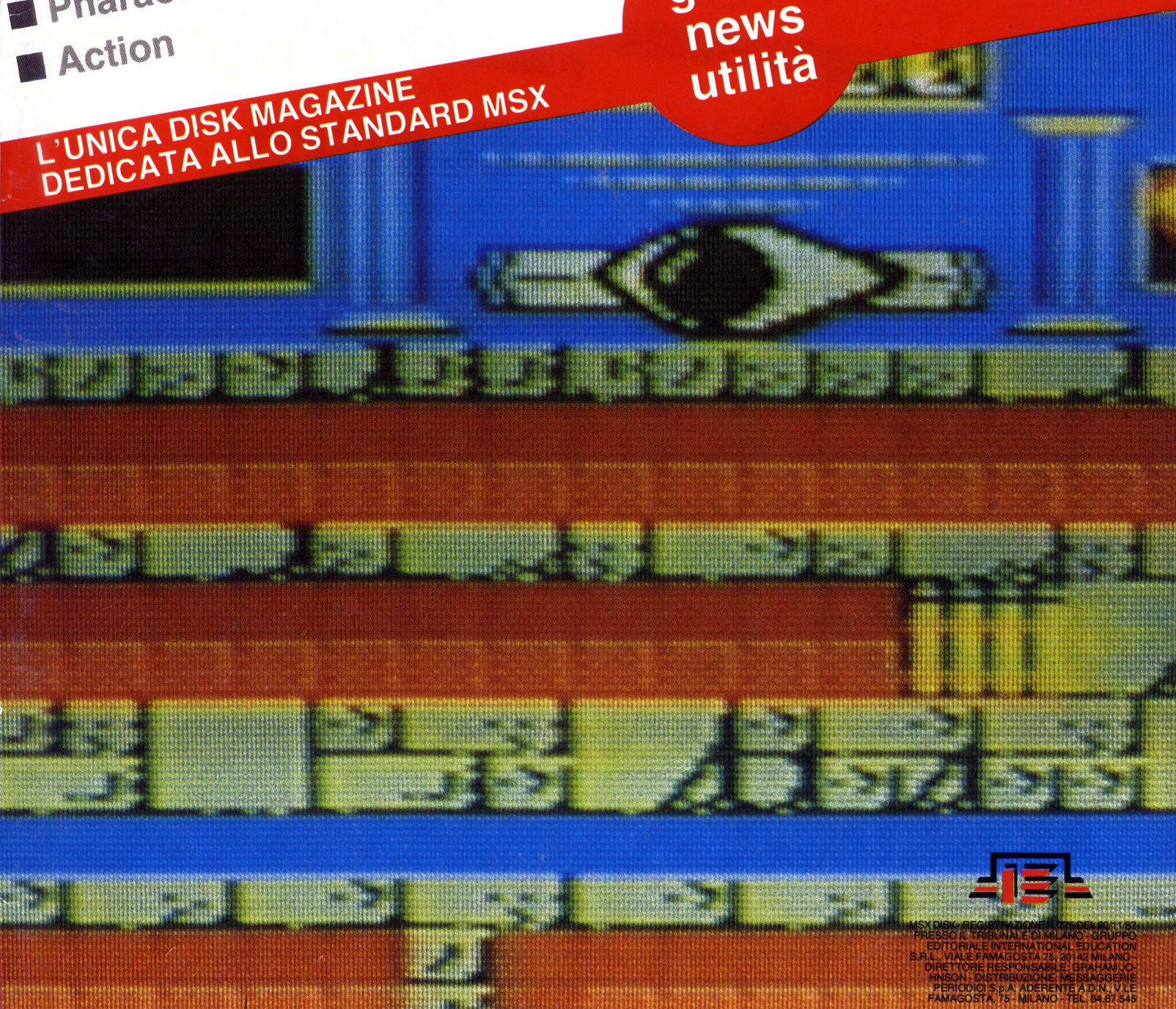
N 21

- Golvellius
- G-Beret
- Pharaons
- Action
- Graphos
- Tasword
- Finance Plus 4

£ 14.000

giochi
news
utilità

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.I. V.LE
FAMAGOSTA, 75 - MILANO - TEL. 84.67.545

# MSX DISK

## SOMMARIO



## SUL DISCO

1 Golvellius
2 G-Beret
3 Pharaons
4 Action
5 Graphos
6 Tasword
7 Finance Plus 4

## 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

Voglio dedicare questo editoriale agli utenti di AMIGA e di MS-DOS Compatibili con drive da tre pollici e mezzo. In special modo a chi compra MSX DISK perché pensa che vada bene per il suo computer o perché l'edicolante - che è sicuramente un esperto del settore - dice che va bene.
Mi spiace sentire persone deluse al telefono e cerco sempre - anche se non dovrei - di sostituire come posso i dischetti comprati per sbaglio con qualcosa di valido, almeno con i PC.
Forse chi compra abitualmente questa rivista avrà notato che la copertina è cambiata, non solo nell'immagine grafica ma anche nella scritta che dice "L'unica disk magazine dedicata allo standard MSX".
Ma forse non è chiara abbastanza, forse sarebbe meglio "Non compratela per AMIGA e MS-DOS Compatibili..."


Ma bando alle "ciance"! Come sempre anche questa volta troverete delle novità, più precisamente delle recensioni di giochi, un articolo interamente dedicato alle Simulazioni e poi gli appuntamenti ormai consueti: il Basic, Dentro l'MSX, il Pascal, la Posta, il Libro del Mese, Computer & Programmazione e i nostri programmi. Buon Divertimento & Buon Lavoro!!!

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.**

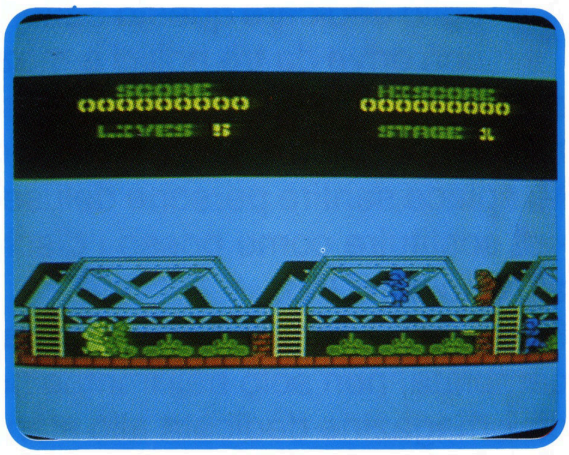
**Ritagliare e spedire a Gruppo Editoriale International Education – Viale Famagosta 75 – 20142 Milano.**

# GOLVELLIUS


Come è ormai una consuetudine, anche questo mese vi proponiamo un gioco per MSX 2 che occupa gran parte del disco ma che in compenso offre grafica e azione super. In questo gioco vestite i panni dell'eroico, alto, bello e biondo Golvellius. Guardacaso, la vostra bellissima ragazza Nice è stata rapita dal mago cattivo di turno che l'ha segregata nel suo castello. Per pura combinazione il castello è un vero e proprio labirinto popolato da mostri di ogni tipo: trolls,parrucchini, fattuchiere, maghi, gnomi e Pippi Baudi. Non vi resta altro che sfoderare la possente spada e correre fra le fetide stanze del mega-castello alla ricerca del vostro amore (forse) immortale. Vestiti i panni di Golvellius, dovrete affrontare i crudelissimi mostri alla ricerca degli indizi più disparati fino a che non trovate il megadirettore della situazione e lo fate fuori con la vostra spada. Sul suo cammino il nostro Golvellius potrà trovare nuova energia ma anche gioielli e preziosi vari che abbondano sempre in giochi di questo tipo. Sul suo cammino troverà anche nuove armi che dovranno essere usate con i mostri più incattiviti. Raggiunta la bella Nice dovrete difenderla strenuamente fino a che non avrete raggiunto entrambe la libertà.

**COMANDI**
Tasti:
[CURSORI] = Movimenti & Salto
[SPAZIO] = Fuoco & Inizio gioco
Joystick in porta 1

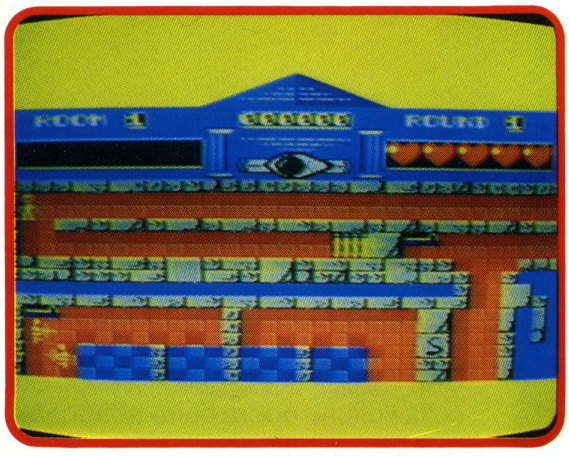
# G-BERET



Sei il protagonista di un'impresa quasi al limite del suicidio dato che dovrai affrontare da solo centinaia di nemici armati fino ai denti. La missione è certamente una delle più pericolose della tua lunga carriera di marines. Sarai infatti impegnato in continui corpo a corpo con i nemici. Per fortuna, di tanto in tanto, c'è la possibilità di sottrarre un'arma ad uno dei nemici così da poterla utilizzare nei momenti di maggior pericolo. I nemici indossano divise di diversi colori. Una divisa bianca indica che il personaggio è un comandante. La divisa verde significa che il soldato è esperto in armi marziali; le divise marroni sono indossate dai soldati semplici. I soldati con le divise blu sono quelli da evitare perché sono armati e sparano a vista. Il gioco prevede due livelli di difficoltà. Nel primo livello dovrai utilizzare le rampe e le piattaforme per sfuggire agli attacchi dei nemici. Fai particolare attenzione agli esperti di karate poiché tenteranno in tutti i modi di eliminarti. Usa le tue doti acrobatiche per scansarli. Sorveglia attentamente i comandanti: quando ne vedrai comparire uno, cerca di ucciderlo, perché solamente in questo modo potrai entrare in possesso del lanciafiamme. Alla fine del ponte si trova un temibile avversario, l'artigliere addetto ai mortai. Usa il lanciafiamme per eliminarlo. Se alla fine di tutte queste traversie sarai ancora vivo sentirai il suono di una sirena che annuncia l'arrivo di autocarri carichi di truppe nemiche.
Anche in questo caso l'uso del lanciafiamme sarà particolarmente utile. Nel caso in cui l'arma fosse scarica avvicinati agli autocarri, impugna la baionetta e colpisci con decisione. Alla fine del primo livello in prossimità della base missilistica troverai un campo minato. Uomo avvisato... Giunto al secondo livello compariranno nuovi tipi di avversari. Quando incontrerai il comandante (sempre con la divisa bianca) non esitare a colpirlo per rubargli prontamente il "bazooka". In questo livello, oltre alla truppa che ti assalta, dovrai vedertela con paracadutisti, feroci cani doberman e vari ostacoli del genere. L'obiettivo finale è quello di liberare i tuoi compagni prigionieri. Il nostro marines è facilmente manovrabile facendo uso del joystick: può correre, strisciare, saltare e lanciarsi all'assalto con la baionetta o con le diverse armi che troverà sul suo cammino.

**COMANDI**
Tasti:
[CURSORI] = Movimenti & Salto
[SPAZIO] = Inizio gioco / Usa baionetta
[SHIFT] = Per sparare o fare a pugni

# PHARAONS



Sulle orme del mitico Indiana Jones, anche voi diventate archeologi e, dopo aver raccolto tutto il coraggio di cui disponete, vi lanciate all'interno della piramide maledetta alla ricerca del preziosissimo tesoro del faraone intestatario della tomba. Ovviamente voi non siete così abili e fortunati come il vostro idolo, e come per incanto risvegliate l'anima del faraone che, grazie ad una potente maledizione, vi seguira col suo grande occhio fino a farvi eliminare dalle mummie e dai fantasmi che si generano all'improvviso. Ovviamente non è tutto qui: qua e là troverete delle vere e proprie trappole come lance nascoste, tagliole e generatori di incantesimi che vi colpiranno con i fulmini magici.
Il labirinto, poi, non sarà libero e percorribile senza problemi ma ad ostacolarvi troverete delle grosse porte che potrete aprire solo con le chiavi sparse per i corridoi. Per sopravvivere potrete scappare oppure difendervi con la vostra pistola che, combinazione, è senza colpi! Per rifornirvi di colpi dovrete usare i caricatori sparsi un po' ovunque nel labirinto ma questi hanno solo sei colpi ognuno e contro tutte quelle mummie sarà davvero un'impresa! Buona fortuna...

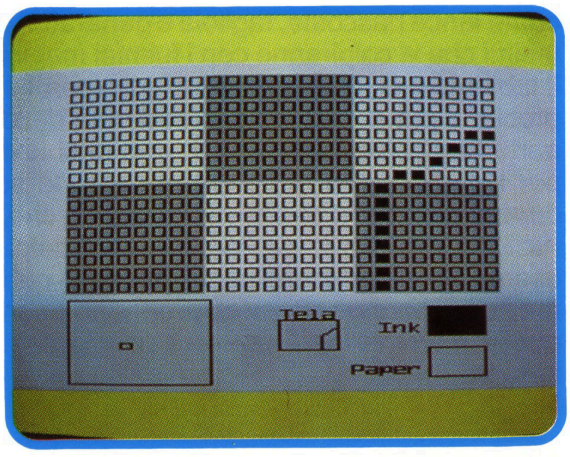
**COMANDI**
Tasti: [CURSORI] = Movimenti
[SPAZIO] = Inizio gioco / Fuoco
Joystick in porta 1

# ACTION


Sicuramente non siete Sean Connery o Roger Moore ma questa volta James Bond dovete interpretarlo voi! La vostra missione, questa volta, è di penetrare in un grattacielo di 30 piani e rubare tutti i documenti preziosi che vi sono contenuti. Ovviamente non entrate dal portone principale ma, grazie ad un resistente cavo, arrivate sul tetto per "via aerea". Da qui dovete raggiungere il piano seminterrato dove troverete una potènte macchina necessaria alla fuga. Grazie ai numerosi ascensori e alle scale mobili potrete passare agilmente da un piano all'altro alla ricerca degli uffici che contengono i documenti preziosi. Gli uffici sono molti ma riconoscere quelli che vi interessano sarà facile: sono quelli con la porta di colore diverso e con lo zerbino. Per entrarvi dovrete passare sullo zerbino dirigendovi verso la porta. Abbastanza simile è la procedura per attivare le scale mobili: dovrete posizionarvi sulle piattaforme di accensione e quindi premere il tasto del cursore Alto o Basso a seconda di dove dovete andare. Sempre con i cursori controllate la direzione degli ascensori, ma solo se ci siete dentro non sopra. Sempre con i cursori, o con il joystick, vi muovete a destra e a sinistra e potete abbassarvi per schivare i colpi mentre per saltare dovrete usare il tasto [Z]. Il palazzo è infestato da agenti segreti nemici che sbucheranno dagli uffici: potete spararli grazie alla vostra licenza di uccidere, ma se li abbattete sparando ai lampadari riceverete dei punti bonus. Una volta "visitati" gli uffici interessanti dovrete raggiungere il garage e rubare un'automobile. Per far cominciare il gioco dovrete attendere la fine della presentazione iniziale e la comparsa della scritta "TECLA ESPACIO".

**COMANDI**
Tasti: [CURSORI] = Movimenti
[SPAZIO] = Inizio gioco – [Z] = Fuoco
[X] = Salto – Joystick in porta 1

# GRAPHOS




In molti ci hanno richieste utility grafiche e sembra proprio che quelle pubblicate fino ad oggi non siano davvero sufficienti.
Così, anche questa volta, vi proponiamo un programma dedicato alla grafica molto semplice da usare e che, rispetto al passato, presenta dei menù invece delle consuete icone.
Pur non essendo in Italiano, i menù sono abbastanza comprensibili e le diverse opzioni sono facilmente selezionabili: dovrete semplicemente posizionare la barra sull'opzione desiderata tramite i tasti del cursore e poi premere il tasto di Invio.
E' possibile usare un joystick ma deve essere collegato alla porta 2 invece della consueta porta 1.

Ecco il menù principale e le funzioni associate alle diverse opzioni:

## DISPLAY TELA

Serve per vedere la "tela" in memoria senza modificarla. Dopo la selezione di questa opzione viene mostrato un ulteriore menù di quattro opzioni.
Queste opzioni fanno apparire la tela su schermo in quattro diversi modi.
Per tornare al menù principale si deve usare il tasto [ESC].

## EDITA TELA

Questa opzione permette di creare o modificare la tela in memoria. Per tornare al menù si deve premere il tasto di Invio.
Il cursore si sposta con i soliti tasti con le frecce mentre per disegnare dovete premere lo spazio.

Usando i tasti delle frecce mentre premete lo [SHIFT] la velocità di spostamente del cursore aumenta notevolmente.

Con il tasto [GRAPH] si cambia il colore del cursore grafico, mentre con i tasti funzione si accede alle opzioni di edit.

Con l'[ESC] si esegue l'UNDO, cioè si cancella l'ultima operazione effettuata.

Con il tasto [CLS] si esegue un'operazione simile cancellando l'ultima cosa che è stata disegnata.

Col tasto [SELECT] si possono selezionare gli attributi, cioè il colore dell'inchiostro e dello sfondo, e si ritorna al disegno con il tasto di Invio.

Con il tasto [F1] si accede al menù DISEGNO che comprende 9 opzioni relative al disegno quali riempimento, cerchi, rettangoli, linee, tracce, spray, raggi, blocchi, etc.

Si torna al disegno selezionando l'opzione scelta quando si preme il tasto di Invio, mentre con [ESC] si torna al disegno senza scegliere alcuna opzione.

Con [F2] si seleziona il menù TESTO composto da sei opzioni: Normal, Italic, Bold, Doppio, Doppio Bold e Largo.
Queste opzioni corrispondono a diversi stili dei caratteri.
Selezionando un'opzione con il tasto di Invio si torna al disegno e si può inserire del testo nel disegno.

Con [ESC] si torna al disegno senza scegliere alcuna opzione.

Con [F3] si accede al menù TELA che permette di: Salvare la tela, Invertire il video, Invertire gli attributi, Togliere il video, Togliere gli attributi, Rimettere il video, Rimettere gli attributi, Cancellare la tela e Stamparla.

Anche qui si seleziona l'opzione con il tasto di Invio e si torna al disegno senza selezione tramite il tasto [ESC].

Con [F4] si seleziona il menu ADJUST che permette: Scroll del video pixel per pixel; Scroll byte per byte; Rotazione pixel per

# GRAPHOS

per pixel; Scroll byte per byte; Rotazione pixel per pixel; Rotazione byte per byte.
Invio seleziona l'opzione scelta, [ESC] riporta al disegno.

Il tasto [F5] porta ad un menù MISTO che contiene:

Zoom della zona di schermo desiderata,
Funzione Shape,
Scroll della zona desiderata,
Attivazione griglia video.

## ARCHIVIAZIONE TELA
Permette di archiviare la tela creata.
Dopo la selezione si deve indicare cosa salvare (Display, Layout, Compact o Editor), e poi il supporto magnetico (Disco o Cassetta).
L'operazione si può interrompere tramite il tasto [ESC].

## RECUPERA TELA
E' esattamente il contrario dell'operazione precedente e serve per leggere i dati precedentemente registrati. Selezionando Display potete provare a leggere il file SHOW2 che è contenuto nel disco.

## EDITA ALFABETO
Questa opzione permette di personalizzare i caratteri testo del programma.
Una volta selezionata l'opzione è possibile scegliere il carattere da disegnare o modificare tramite i tasti del cursore.
La selezione avviene tramite il tasto di Invio che poi porta all'edit del carattere.
Il carattere si disegna muovendo il cursore con i soliti tasti e accendendo o spegnendo i pixel con lo spazio.
Dall'edit del carattere si può uscire in due modi: con l'Invio, che salva il nuovo carattere in memoria, o con l'[ESC] che non modifica il carattere originale.
Si torna al menù principale tramite il tasto [ESC].
Con il tasto [SELECT] si attiva il menù funzioni che offre cinque diverse opzioni: Pulizia griglia carattere, Inversione griglia, Specchio, Rotazione, Edit P1 (Parte schermo).

## ARCHIVIA ALFABETO
Similarmente all'opzione di archiviazione della tela, questa permette di salvare, su Disco o Cassetta, l'alfabeto creato tramite l'opzione precedente.
Con [ESC] si torna al menù principale mentre con l'Invio si scelgono le opzioni offerte.

## RECUPERA ALFABETO
Legge da Disco o Cassetta un'alfabeto precedentemente creato. [ESC] riporta al menù principale.

## CREA SHAPES
Tramite questa opzione si accede ad un sottomenù composto da altre cinque opzioni: le prime quattro opzioni permettono di creare quattro diversi tipi di Shapes mentre l'ultima opzione cancella tutti gli Shapes in memoria.
Gli Shapes sono delle porzioni di schermo che possono essere rielaborate e poi richiamate a proprio piacimento nella fase di edit, come fossero dei grossi sprites riutilizzabili all'infinito, un po' come dei timbri.
Con [ESC] si torna al menù principale mentre l'editor degli Shapes sfrutta i soliti tasti, Cursori e Invio.

## ARCHIVIA SHAPES
Similarmente alle precedenti opzioni di Archiviazione questa serve per archiviare gli Shapes.
Il solito tasto [ESC] riporta la selezione al menù principale.

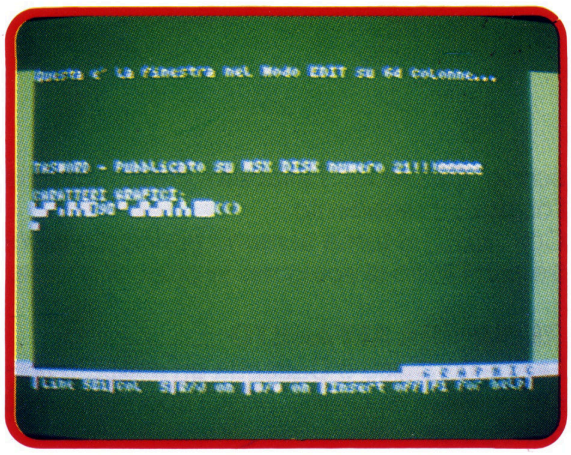
## RECUPERA SHAPES
Permette di leggere, da Disco o Cassetta, gli Shapes precedentemente registrati.
[D] o [C] per selezionare Disco o Cassetta.
Con [ESC] si torna al menù precedente.

## DIRECTORY
Permette di vedere la lista dei files nel dischetto selezionato.

## VERSIONE SISTEMA
Mostra i dati relativi al programma, cioè autore e versione del programma.

## BASIC
Riporta l'esecuzione all'MSX BASIC.
Dopo la selezione di questa opzione è necessario confermare l'abbandono del programma premendo [S] per andare i Basic o [N] per tornare al menù principale.
Una volta in Basic è possibile riattivare il programma premendo il tasto funzione [F1].

# TASWORD



In molti ci hanno scritto riguardo al word processor che abbiamo pubblicato negli scorsi numeri soprattutto per un motivo: il programma funziona su tutti i computer MSX ma lavora bene solo sugli MSX 2 che sono ad 40 colonne.

Abbiamo così deciso di proporvi un programma più completo che, come vedrete, non necessita dell'MSX DOS e permette di lavorare con più di 40 colonne anche con il meno potente degli MSX 1.

Il programma è decisamente più lento del WordStar pubblicato alcuni numeri fà, ma permette di lavorare sia su 40 che su 64 colonne grazie ad un metodo efficace che sfrutta il modo grafico su 32 colonne.

Praticamente il programma fa uso di un seti di caratteri composto da una matrice di 4 x 8 pixels invece di quella di 8 x 8, raddoppiando le colonne utilizzabili.

Ma veniamo ai comandi disponibili con
**TASWORD**
nel modo EDIT:

[F1] = Pagina di Aiuto.

[RETURN] = Ritorna alla fase di Edit quando ci si trova nelle pagine di Aiuto.

[F2] = Riformatta il Paragrafo.

[F3] = Cancella la linea su cui si trova il cursore.

[GRAPH] = Seleziona il modo GRAPHIC.

In questo modo si potranno immettere caratteri grafici semplicemente premendo i tasti dall'1 all'8 con o senza la pressione contemporanea del tasto [SHIFT].
I caratteri grafici disponibili sono compatibili con le stampanti Epson.

[CURSORI] = Movimenti cursore.

[CTRL][Cursore SU] = Scroll del testo verso l'alto.

[CTRL][Cursore GIU'] = Scroll del testo verso il basso.

[CTRL][Cursore SINISTRA] = Sposta il cursore alla prima parola a sinistra.

[CTRL][Cursore DESTRA] = Sposta il cursore alla prima parola a destra.

[SHIFT][Cursore SU] = Porta il cursore all'inizio del testo.

[SHIFT][Cursore GIU] = Porta il cursore alla fine del testo.

[SHIFT][Cursore SIN] = Muove la linea verso sinistra.

[SHIFT][Cursore DES] = Muove la linea verso destra.

[HOME] = Centra la riga selezionata

[DEL] o [BS] = Cancella il carattere alla sinistra del cursore e lo sposta.

[CTRL][STOP] = Passa al menù principale.

[CTRL][B] = Marca l'inizio del Blocco.

[CTRL][V] = Marca la fine del Blocco.

# TASWORD


[CTRL][Q] = Muove il Blocco marcato.

[CTRL][N] = Copia il Blocco marcato.

[CTRL][E] = Attiva/Disattiva la Giustificazione a Destra.

[CTRL][W] = Attiva/Disattiva il Word-Wrap.

[CTRL][J] = Giustifica Linea.

[CTRL][U] = DeGiustifica Linea.

[CTRL][T] = Cambia modo video 40/64.
Se ci si trova nel modo video a 64 colonne la visualizzazione del testo passa alle 40 colonne con scroll orizzontale.
Viceversa, se si è nel modo a 40 colonne la finestra del testo passa alle 64.

[CTRL][Y] = Cancella il testo inserito.
Dopo aver dato questo comando si deve confermare la cancellazione del testo in memoria premendo [Y] per Sì, altrimenti con [N] come No si torna alla fase di Editing.

[CTRL][O] = Ricerca o rimpiazza la parola inserita. Se si inserisce solo la parola e nessuna parola da sostituirgli, il cursore viene posizionato sulla prima parola trovata altrimenti tutte le parole uguali a quella indicata vengono sostituite con la seconda parola inserita.

[CTRL][Z] = Attiva/Disattiva modo Insert.

[CTRL][A] = Fissa margine Sinistro.

[CTRL][S] = Azzera i margini.

[CTRL][D] = Fissa il margine Destro.

[CTRL][F] = Scroll veloce verso il basso.

[CTRL][G] = Scroll veloce verso l'alto.

Dopo aver visto i comandi della fase di Editing, i seguenti tasti rappresentano i comandi presenti nel MENU' PRINCIPALE a cui si accede dalla fase di Editing tramite la combinazione di tasti [CTRL][STOP]:

[P] = Stampa il testo.

[S] = Salva il testo su disco.

[L] = Carica il testo dal dischetto.

[M] = Aggiunge al testo in memoria un altro testo caricandolo dal disco.

[V] = Verifica il salvataggio di un testo.

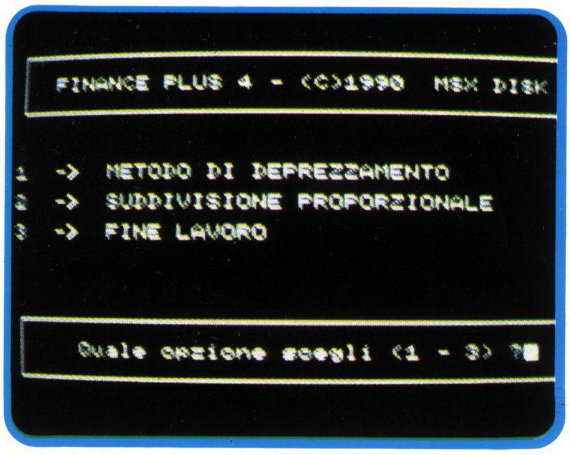
[R] = Passa alla fase di Edit.

[D] = Definizione del set di caratteri di controllo della stampante.

[T] = Salva il programma TASWORD con le modifiche effettuate, cioè margini, set di caratteri di controllo, giustificazione, finestra da 40 o 64, word-wrap, etc.

[B] = Riporta l'esecuzione all'MSX BASIC.
Una volta in Basic si potrà far ripartire il programma semplicemente digitando RUN seguito dalla pressione del tasto di Invio.

# FINANCE PLUS 4




Continuando la serie Finance Plus vi ricordo quello che è stato già detto in precedenza: questi programmi hanno il duplice scopo di fornire un aiuto nell'ambito finanziario e offrire uno spunto didattico ai principianti che entrano nel mondo della programmazione in Basic.
Per utilizzare il programma sarà sufficiente premere il tasto [7] quando compare il menù principale di Msx Disk. Il file 7.BAS è il file sorgente pronto per essere esaminato e rielaborato.
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 - METODO DIDEPREZZAMENTO;
2 - SUDDIVISIONE PROPORZIONALE;
3 - FINE LAVORO.
Nel file sorgente potrete trovare che la prima opzione viene espletata dalla parte che va dalla linea 1000 alla linea 1350 mentre l'analisi degli investimenti comuni è compresa tra la linea 2000 e la linea 1310. Come sempre, dalla linea 3000 in poi trovano posto le routine di servizio.

## METODO DI DEPREZZAMENTO

Il metodo di deprezzamento accelerato fornisce un deprezzamento maggiore nei primi anni, ma ad un certo punto è più realistico passare ad un deprezzamento lineare, cosicché si ottiene negli ultimi anni un deprezzamento maggiore di quello che si avrebbe con l'altro metodo (deprezzamento accelerato).
Il calcolo è condotto considerando un costo fisso di un milione.
Il costo effettivo dell'articolo non ha nessuna importanza: il valore di un milione permette semplicemente di effettuare calcoli più precisi.
Utilizzando questa parte del programma, dovrete inserire il metodo di deprezzamento valutato per l'articolo in percentuale (per esempio 125, 150, 200 e così via), la vita utile dell'articolo in anni, il numero di mesi di deprezzamento per il primo anno di vita utile (un anno intero sarà inserito come 12 mesi).

Esempio
Una ditta acquista una macchina per stampaggio di materie plastiche che ha una vita utile di cinque anni. Per l'anno fiscale presente, rimangono sei mesi di deprezzamento e la ditta adotta un sistema di deprezzamento accelerato del 200%.
Quando conviene passare dal metodo accelerato a quello lineare per massimizzare l'ammontare della somma deprezzata?

Risposta: la ditta deve passare al deprezzamento lineare al quinto anno.

## SUDDIVISIONE PROPORZIONALE

Questo programma suddivide una quantità data, nella proporzione in cui ogni singolo numero di un gruppo sta alla somma di tutti i numeri del gruppo dato. Per prima cosa il programma richiede di introdurre la precisione, espressa come numero di posti decimali, fino ad un massimo di 9.
Quindi, si introduce il totale da suddividere proporzionalmente e il numero di parti in cui tale somma deve essere suddivisa.
A questo punto si introducono le quantità da assumere come base per la suddivisione.
Il programma mostra una tabella che riporta ciascuna di queste quantità, la percentuale rispetto alla somma di tali quantità, e la corrispondente porzione della quantità da suddividere.
Al termine vengono visualizzati i totali di queste tre colonne.

Esempio
Un candidato alle elezioni vuole suddividere gli addetti alla campagna elettorale tra i 6 distretti della città basandosi sulla popolazione di ogni distretto.
Egli può disporre di 42 addetti, e la popolazione è distribuita nei sei distretti come segue :

Distretto 1: 29.842 abitanti;
Distretto 2: 17.420 abitanti;
Distretto 3: 14.625 abitanti;
Distretto 4: 24.314 abitanti;
Distretto 5: 21.209 abitanti;
Distretto 6: 18.956 abitanti.

Risposta:
Distretto 1 = 10 addetti;
Distretto 2 = 6 addetti;
Distretto 3 = 5 addetti;
Distretto 4 = 8 addetti;
Distretto 5 = 7 addetti;
Distretto 6 = 6 addetti.

# IL LIBRO DEL MESE


**MSX LINGUAGGIO E PROGRAMMAZIONE**
*di Ekkehard KAIER*
*Edito da Tecniche Nuove*
*Pagine: 372*
*Prezzo: 43.000 lire*

Questa volta vi proponiamo un libro dalle dimensioni decisamente inconsuete per un libro sull'MSX e dal contenuto interessantissimo.

Il libro è diviso in più parti indipendenti tra loro. Ovviamente, la prima parte è di introduzione all'informatica e può essere saltata a pie' pari da chi è già esperto.

Questa prima parte è davvero ottima e introduce molto bene i diversi concetti fondamentali, spiegando cosa sono hardware, software e firmware, cosa sono i grossi calcolatori e i microcomputer, le strutture di dati e le strutture di programma, i sistemi operativi e i programmi applicativi, come si installa un programma già pronto e come implica programmare.

Dopo la lettura di questa parte si è messi in grado di inquadrare i computer MSX nel panorama informatico.

Il manuale continua poi con l'MSX spiegando come si usano tastiera, monitor, floppu, drive e stampanti MSX, come si scrive il primo programma e quali comandi contiene l'MSX BASIC, qual'è la differenza tra lavoro a livello di MSX-BASIC e al livello del sistema operativo MSX-DOS.

Spiega poi i comandi dell'MSX-DOS con chiari esempi come viene fatto con la spiegazione dei comandi MSX-BASIC.

Dopo la lettura di questa seconda sezione si è in grado di usare davvero il computer MSX laciando programmi e scrivendone di semplici in BASIC con la possibilità di memorizzarli.

Dopo queste due eccellenti sezioni, il manuale continua con un vero e proprio corso di programmazione che riguarda programmi che illustrano:

- le varie strutture (sequenziali, condizionate, iterative, sottoprogrammi);
- l'elaborazione di testi e l'input/output;
- la programmazione in codice macchina e la ricerca, ordinamento e classificazione di dati;
- i file sequenziali e quelli ad accesso diretto;
- la grafica, gli sprite e gli effetti sonori.

Il libro è decisamente eccellente e contiene 101 programmi esemplificativi riportati e commentati accuratamente.

I capitoli 2 e 3 si completano logicamente e devono essere letti in ordine di successione, mentre la lettura del capitolo 1 può procedere parallelamente.

In conclusione, questo libro è da non perdere: è adattassimo ai principianti, mentre per gli utenti più esperti è un riferimento indispensabile grazie soprattutto agli indici analitici e alle accurate tabelle riassuntive.

# POSTA


Spett. MSX DISK, sono un grande affezionato alla vostra mega rivista e vi chiedo se nel prossimo numero di MSX DISK ci fossero i seguenti giochi: Berretti Verdi; Carrier Command; Thundercats; Yes, prime minister for you; MASK; Xybots e infine un programma per trasportare i giochi da cassetta a disco.
Per migliorare la vostra rivista vi consiglio di mettere la rubrica IL MERCATO DI HARDWARE E SOFTWARE; mettere più pagine di recensione giochi rappresentando la schermata del gioco; mettere anche delle pagine dove vengono pubblicati listati di giochi o programmi vari.
Infine vi chiedo di stampare nella copertina le schermate dei giochi spiegando in modo sintetico di che cosa si tratta.
(Continuate così che siete forti!!!)

**Andrea MANUZZATO - Breganze (VI)**

*Prima di tutto voglio ringraziare il nostro giovane amico per gli splendidi disegni che ci hai inviato: sono davvero belli.*
*Per quento riguarda i consigli che ci dai, come vedi stiamo migliorando sempre più e già da questo numero pubblichiamo delle nuove recensioni e c'è il gioco Green Beret (Berretti Verdi) che ci hai chiesto.*
*Per quanto riguarda gli altri giochi, non tutti esistono in versione MSX: l'unico che abbiamo visto è Xybots, ma non perdere le speranze e mandaci altri consigli e disegni.*

Gentile redazione di MSX-DISK vi scrivo con un po' di ritardo, mo solo oggi ho deciso di mettere ordine nei miei dischetti (che sono quelli allegati alla vostra preziosa rivista).....devo chiedervi delle informazioni su alcuni programmi che non girano sul mio Msx:

1 - ROCK-TOWN pubblicato su MSX DISK numero 13 e COLOSSUS pubblicato sul numero 18 non li riesco a caricare: compare la videata di presentazione iniziale, poi il drive comincia a funzionare in modo anomalo.
Ho provato a caricarli da Basic previa una poke -1,170 e il risultato è un Sintax Error in 40 in un listato "impossibile".
GO ALONE invece gira regolarmente.
E' un problema di incompatibilità o è il dischetto difettoso?

2 - Su MSX numero 16 compare sul menù principale l'opzione 9 per il caricamento di Valley; però premendo 9 non succede assolutamente niente; la videata resta sullo schermo e il computer non da segni di vita.
Da Basic invece il caricamento riesce ma non posso giocare perché dopo il run compare un messaggio di Sintax Error in 332 (a parte vi listo la linea incriminata).

Per concludere, vi saluto aspettando una vostra rispota (spero di essere stato chiaro) e presentandovi il mio computer: è un Msx 1 TOSHIBA HX 10 con drive FENNER DD 300 e stampante COMMODORE MPS 1230.
Cordiali saluti e continuate così.

**Paolo AGRIMI - Treviso**

*Cominciamo dal punto 1: si tratta proprio di problemi di incompatibilità.*
*Infatti, lei stesso ci dice che GO ALONE funziona senza problemi, mentre molti lettori ci hanno chiamato per dirci che sulla loro macchina il programma non funziona.*
*Anche COLOSSUS ha dato problemi, mentre nessuno ci ha chiamato o scritto per ROCK-TOWN.*
*E' ancora il dannato problema della non perfetta compatibilità che continua a farci e farvi soffrire.*
*Purtroppo non c'è soluzione e noi, d'altronde, non possiamo provare tutto il software su tutti gli Msx esistenti.*
*Per quanto riguarda il secondo punto, già in precedenza ho risposto ad una domanda simile.*
*Il problema è che VALLEY è un programma troppo lungo e con alcuni MSX non vuol saperne di girare quando è eseguito dal disco: funziona solo da cassetta!*
*Per questo abbiamo approntato una nuova versione del numero 16 di MSX-DISK con un'altra avventura funzionante al posto di VALLEY.*

# POSTA

*Per averlo è sufficiente inviarci una copia di MSX-DISK 16: il dischetto vi verrà rispedito con le spese a nostro carico.*

Spett. redazione di MSX DISK, Vi scrivo a proposito del programma VALLEY.BAS da voi inserito nel dischetto della rivista n. 16 e menzionato nella n.17 a proposito di una sua malfunzione.
Potrei scrivervi anch'io in effetti, come altri lettori, per il fatto che tale programma come era registrato sul dischetto non funzionava perfettamente, ma poi ho risolto il problema: ho analizzato il listato ed ho trovato un piccolo "bug".
Il problema che dava a me (che non può avere niente a che vedere con problemi di incompatibilità fra MSX) era quello di non riuscire nemmeno ad autoinizializzarsi, poiché dava errore nel caricamento dei DATA, caricamento effettuato appunto in fase iniziale.
Ho quindi studiato attentamente il listato ed ho visto che mancava nei DATA la stringa necessaria ("FD") che consentiva al sistema di capire frasi, che non doveva più leggere i dati sui verbi e gli oggetti ma sulle frasi che descrivono i luoghi (e cambiare in sostanza la READ da effettuare).
Il programma quindi si può FACILMENTE correggere inserendo la linea mancante una volta effettuato il RENUM:

digitate 2325 DATA "FD"

Ed ora il programma girerà senza problemi.
Distinti Saluti...

**Gianluca PISANO - Livorno**

*Desidero ringraziarla per il suo utile consiglio che purtroppo non posso verificare prima di scrivere questa risposta.*
*L'unica cosa che posso fare è invitare i lettori a provare questa correzione per vedere se funziona sui loro sistemi.*
*Comunque, io credo che sia una cosa abbastanza soggettiva e limitata solo ad alcuni MSX.*
*Provare per credere!*

Spett. GRUPPO...,
Prego esaminare la possibilità di inviarmi il n. 16 di MSX DISK (completo di dischetto e manualetto) con pagamento contrassegno o altra forma che cortesemente vogliate indicarmi.
Con la presente chiedo anche se esiste o avete già prodotto o comunque dove si possa reperire un compilatore PASCAL o TURBO PASCAL per MSX e quali prezzi sono eventualmente applicati.
Ringraziando, porgo distinti saluti e i complimenti per la vostra rivista che da poco ho scoperto.

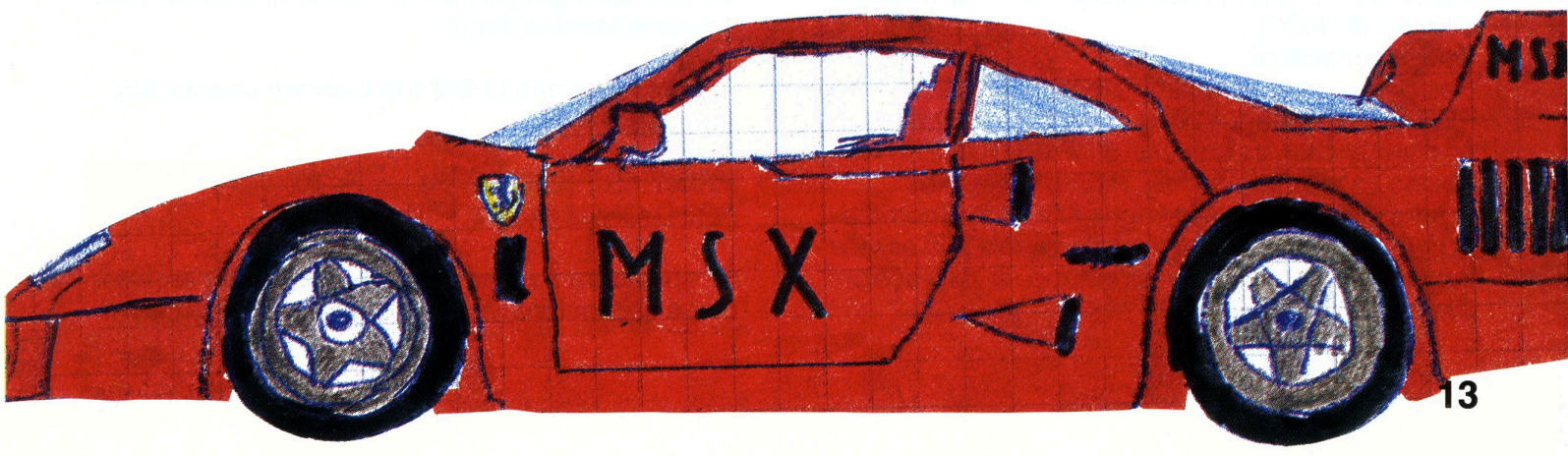
**Franco D'ANDREA - Chiavari (GE)**

*Sembra incredibile, ma quando in redazione arrivano copie degli ultimi numeri di MSX DISK stampati, queste vanno subito esaurite come arretrati.*
*Così non possiamo accontentarla, ma se le interessa può procurarsi una copia del dischetto con la fotocopia del relativo manualetto all'MSX CLUB ITALIA di Lodi ad un prezzo di 7000 lire.*
*Anche il compilatore TURBO PASCAL le potrà essere fornito dal club di Lodi, ad un prezzo non superiore a 10-15.000 lire manuale escluso.*

*L'indirizzo è:*

*MSX CLUB ITALIA - Cp 34*
*20075 Lodi Centro (MI).*

**MSX DISK!**



# BASIC (Parte VIª)

## VARIABILI STRINGA E CODICE ASCII

Una variabile numerica, come ben sapiamo, può contenere un valore numerico.
Una VARIABILE STRINGA (riconoscibile perché il nome termina con $ dollaro) contiene invece un certo numero di caratteri (anche nessuno, nel caso della stringa nulla) generalmente in codice ASCII.
Ad ogni carattere, perciò, è associato un numero: il valore del codice ASCII corrispondente.
Tuttavia, numeri e stringhe sono due cose distinte. I numeri possono essere usati in espressioni, mentre le stringhe sono "ripetute" dal calcolatore tali e quali. Su di esse è possibile operare soltanto per mezzo di apposite istruzioni, che non abbiamo ancora trattato.

Ecco alcuni esempi di errori:

```
A="3"   ("3" è un carattere, non un numero)
C$=12   (12 è un numero, non un carattere)
IF A=C$ THEN ..... (numeri e stringhe non sono confrontabili)
```

A volte, però, può essere utile conoscere il codice ASCII di un carattere o, viceversa, produrre un carattere dato il suo codice (ad esempio, per produrre i caratteri di controllo necessari ad alcune unità video).
Esistono, a questo scopo, due funzioni BASIC, denominate ASC (CODE in alcune macchine) e CHR$.
Cominciamo dalla prima.

Ecco un programma che chiede un carattere e stampa il valore del codice ASCII corrispondente:

```
10 INPUT "DAMMI UNA LETTERA: ";A$
20 B=ASC(A$)
30 PRINT "IL CODICE DI ";A$;" E' ";B
40 GOTO 10
```

Alla linea 20 compare la funzione ASC, che calcola il codice ASCII del suo argomento, cioé del carattere contenuto nella stringa A$.
Se la stringa A$ è lunga più di un carattere, ad esempio "ROMA" o "NO", la funzione ASC calcola il codice del primo carattere (cioé la "R" di "ROMA" o la "N" di "NO").
Due osservazioni.
Per prima cosa, abbiamo detto che ASC è una FUNZIONE. Una funzione è un tipo particolare di operatore che, in base ai suoi ARGOMENTI (in questo caso la stringa A$), ritorna un valore.
In altre parole, l'intera funzione ASC(A$) si comporta come un valore numerico e come tale può essere impiegata, ad esempio all'interno di espressioni come 3+ASC(A$).

In secondo luogo, in BASIC il nome di una funzione indica anche il TIPO del risultato: se il nome termina con $ (dollaro), la funzione RESTITUISCE (equivale a) una stringa, altrimenti (è il nostro caso) restituisce un valore numerico.

Proviamo a far girare il programma:

```
RUN

DAMMI UNA LETTERA: A
IL CODICE DI A E' 65
DAMMI UNA LETTERA: G
IL CODICE DI G E' 71
DAMMI UNA LETTERA: M
IL CODICE DI M E' 77
```

e così via fino al [CTRL]-[C] (se il programma sta eseguendo una INPUT, occorre premere [RETURN] dopo il [CTRL]-[C]).
Per evitare di dover terminare un programma con un'interruzione forzata (come [CTRL]-[C]) è preferibile inserire nel programma stesso una condizione che ne determini l'arresto quando lo si desidera.
Possiamo stabilire ad esempio, che il programma esegua un'istruzione END (e quindi si fermi) quando si risponda all'INPUT con un certo carattere.

Aggiungiamo queste due istruzioni:

```
25 IF A$ = "*" GOTO 50
50 END
```

Se, in risposta alla INPUT della linea 10, viene dato il carattere # (DIESIS o CANCELLETTO o POUND), la condizione posta nella IF diventa vera ed il programma salta alla linea 50, dove termina.
Data un funzione, se ne esiste un'altra che compie il lavoro opposto (o RECIPROCO), quest'ultima prende il nome di FUNZIONE INVERSA.
Tornando alla nostra ASC, esiste una funzione inversa che si chiama CHR$.
La CHR$ produce (restituisce) il carattere avente come codice ASCII il numero dato.
Per esempio CHR$(65) corrisponde alla lettera A, CHR$(66) alla B, e così via. Poiché la funzione CHR$ costituisce un carattere (cioé una stringa di un solo carattere) e non un numero, il nome della funzione termina con $.

Per verificare che CHR$ è la funzione inversa ASC, basta scrivere:

```
PRINT ASC (CHR$ (70))      -› 70
PRINT CHR$ (ASC ("F"))     -› F
```

(lo spazio tra il nome della funzione e la parentesi non è obbligatorio, lo mettiamo solo per meglio evi-

denziare il nome stesso).
Vediamo ora due programmi che potete usare per familiarizzarvi con le funzioni ASC e CHR$:

Segretissimo 1 e Segretissimo 2.

Il primo traduce una frase in "codice segreto" (una lettera alla volta), con l'ausilio della funzione ASC.
Il secondo interpreta il "codice segreto" e lo riporta in chiaro, mediante la funzione CHR$.

Ecco il primo programma.

```
10 REM --- SEGRETISSIMO 1 ---
20 PRINT "DAMMI LA CHIAVE SEGRETA (1-37)";
30 INPUT K:IF (K<1) OR (K>37) GOTO 20:REM
CONTROLLO CHIAVE
40 PRINT "SCRIVI UNA LETTERA DA CIFRARE
(A-Z, # PER FINIRE) ";
50 INPUT A$:IF (A$<"A") OR (A$>"Z") GOTO 40:
REM CONTROLLO
60 IF A$="#" THEN END
70 B=ASC(A$)+K
80 PRINT B
90 GOTO 50
```

Qualche commento sul programma.

Le linee 20-30 chiedono la "chiave segreta", che viene assegnata alla variabile K.
La linea 30, in particolare, mostra come si costruisce un INPUT CONTROLLATO, che accetti soltanto i valori voluti (se la chiave non è tra 1 e 37, ripete la domanda alla linea 20).
Le linee 40-50, dove il programma chiede una lettera da cifrare, sono un altro esempio di input controllato.
Il controllo alla linea 50, però, è diverso dai soliti: confronta stringhe invece di numeri.
Un'espressione logica tipo IF A$<"A" è vera se il primo operando (il contenuto di $) precede, in ordine alfabetico (in realtà, in ordine di codici ASCII, ma fa lo stesso), il secondo operando ("A").
Nei computer che non accettano questo genere di operazione, occorre confrontare i codici ASCII.

In questo caso, la linea 50 andrebbe riscritta:

```
50 INPUT A$:IF (ASC(A$)<ASC("A")) OR
(ASC(A$)>ASC("Z")) GOTO 40
```

La linea 60 consente di terminare il lavoro di cifratura: quando si scrive # viene eseguito l'END ed il programma termina.

Se il calcolatore non accetta un END in mezzo al programma, occorre scrivere:

```
60 IF A$="#" GOTO 100

100 END
```

Alla linea 70, finalmente, la lettera viene crittografata, sommando K al valore del suo codice ASCII.
La linea 80 stampa il "cifrato segreto" da trasmettere al "comando operativo".
Per esempio se K = 4, si ottiene una conversione di questo tipo:

```
A   E   R   E   O
69  73  86  73  79
```

Si vede che ogni valore numerico è ottenuto sommando al codice ASCII la chiave

```
K=4(A=65+4=69).
```

Con Segretissimo 2, il messaggio segreto viene decrittografato. Soltando il "comando operativo", che conosce la chiave può effettuare la decodifica.

Ecco Segretissimo 2:

```
10 REM --- SEGRETISSIMO 2 ---
20 PRINT "DAMMI LA CHIAVE SEGRETA (1-37)";
30 INPUT K:IF (K<1) OR (K>37) GOTO 20:REM
CONTROLLO CHIAVE
40 PRINT "SCRIVI UN CODICE DA DECIFRARE
(0 PER FINIRE)"; 50 INPUT C
60 IF C=0 THEN END
70 B$=CHR$(C-K)
80 PRINT B$
90 GOTO 40
```

Nelle linee 20-30 viene chiesta, come al solito, la chiave segreta.
Le linee 40-50 chiedono il codice da decifrare.
La linea 60 termina il lavoro se si è battuto 0.
La linea 70 ricostruisce il carattere originario decrittografando il codice: prima viene sottratto K (le operazioni tra parentesi hanno sempre la precedenza), poi viene assegnato a B$ il carattere avente come codice ASCII il valore calcolato, che non è altro se non il carattere originariamente cifrato con Segretissimo 1.
Questa coppia di programmi, come tutti quelli che presentiamo per illustrare le istruzioni BASIC, ha molte manchevolezze.
Per dirne una, in Segretissimo 2 non c'è alcun controllo nell'input di C, col risultato che il programma può "piantarsi" (terminare con errore) alla linea 60 se C-K non corrisponde ad un codice ASCII valido.
D'altronde, per ora ci interessano la chiarezza e la semplicità degli esempi e non vogliamo appesantirli con troppe rifiniture o confondere inutilmente le idee.

# BASIC (Parte VI[a])

## UTILIZZO DELLA FUNZIONE CHR$

Un tipico campo di impiego della funzione CHR$ è la generazione dei caratteri SPECIALI, o CARATTERI DI CONTROLLO, richiesti da alcune periferiche per compiere determinate funzioni (ad esempio la cancellazione del video).
A volte è possibile inserire questi caratteri tra virgolette, in una normale stringa, ma di solito non risultano poi rileggibili (o appaiono come strani simboli grafici, come nei Commodore), pregiudicando la chiarezza del programma.
Conviene quindi utilizzare la funzione CHR$ per produrre questi caratteri.
Supponiamo, ad esempio, che il carattere con codice ASCII 10 abbia come effetto la cancellazione dello schermo, se stampato con una normale PRINT.

Se desideriamo cancellare lo schermo, basta fare:

```
PRINT CHR$(10);
```

o, meglio ancora, mettere il carattere speciale in una variabile stringa dal nome facilmente ricordabile (CS$ per "cancella schermo") ed usare poi questa nel corso del programma:

```
10 CS$=CHR$(10):REM CANCELLA SCHERMO

140 PRINT CS$;

320 PRINT CS$;
```

Altri caratteri speciali permettono di controllare il movimento del cursore sul video, lo stile usato dalla stampante, ecc.
La funzione CHR$ è anche utile per produrre un carattere che, pur essendo stampabile, non può essere inserito nel testo BASIC: le virgolette (").

Supponiamo di voler stampare una frase contenente le virgolette:

```
10 PRINT "STAMPARE LE VIRGOLETTE (") IN BASIC"
```

Il calcolatore accetterebbe il testo compreso tra le prime due coppie di virgolette come la stringa
"STAMPARE LE VIRGOLETTE
(" e poi segnalerebbe un errore per il resto della frase, considerata fuori dalle virgolette.
Per ovviare a questo inconveniente, alcuni calcolatori (pochi) permettono di introdurre le virgolette battendole due volte.

Nel nostro caso dovremmo scrivere:

```
10 PRINT "STAMPARE LE VIRGOLETTE ("") IN BASIC"
```

Con CHR$ è possibile invece una soluzione valida per qualunque dialetto BASIC.
Dal codice ASCII risulta che le virgolette hanno codice 34.

Possiamo allora scrivere:

```
10 PRINT "STAMPARE LE VIRGOLETTE (";CHR$(34);") IN BASIC"
```

Provando ad eseguire, otteniamo:

```
RUN

STAMPARE LE VIRGOLETTE (") IN BASIC
```

Se le virgolette devono essere usate spesso è meglio assegnarle ad una apposita variabile:

```
10 VI$=CHR$(34):REM VIRGOLETTE

80 PRINT "STAMPARE LE VIRGOLETTE (";VI$;")

IN BASIC "
```

## DATI NEL PROGRAMMA

Le istruzioni DATA, READ e RESTORE consentono di inserire nel programma liste di dati (costanti), che vengono poi rilette in modo analogo ad un INPUT dall'esterno.
E' possibile, ad esempio, avere un'agenda telefonica scritta nel programma stesso, senza dover inserire ogni volta i dati.

## ISTRUZIONI DATA, READ e RESTORE

Spesso è utile INIZIALIZZARE delle variabili, cioè assegnare ad esse dei valori definiti.

Questo può essere agevolmente ottenuto con sequenze di istruzioni del tipo:

```
10 A = 230
20 B = 40
30 C = 155
40 D = 12
50 E = 130
60 F = 18
70 G = 2
80 H = 760
```

L'istruzione DATA consente invece di riunire tutti i dati in un unico punto del programma, di solito all'inizio o alla fine, dove è più leggibile:

```
10 DATA 230, 40, 155, 12, 130, 18, 2, 760
```

L'istruzione READ (leggi) è equivalente alla INPUT, con la differenza che i dati non vengono letti dalla tastiera, ma dalle linee di DATA.

Si può quindi scrivere:

```
120 READ A, B, C, D, E, F, G, H
```

e gli otto valori vengono assegnati alle rispettive variabili.
Se vi sono più linee di DATA, anche in punti diversi del programma, sono lette di seguito come se si trattasse di un'unica linea.
Con le poche nozioni di BASIC finora viste, queste istruzioni non si rivelano di grande utilità.
Possiamo comunque fare un semplice esempio della loro utilizzazione: un'agenda telefonica.
Poiché non abbiamo ancora parlato di come registrare e rileggere dati da nastro o disco, possiamo tenere i dati nel programma stesso, che sappiamo come memorizzare (SAVE) e rileggere (LOAD).

Vediamo il programma.

```
10 REM --- AGENDA TELEFONICA ---
30 INPUT "NOME ? ";NO$:IF NO$="" THEN END
40 RESTORE
50 READ A$, TE$
60 IF A$="FINE AGENDA" THEN PRINT "NON
C'E'.":GOTO 30
70 IF A$<>NO$ GOTO 50
80 PRINT "IL TELEFONO DI ";NO$;" E': ";TE$-
:GOTO 30

1000 REM AGENDA
1010 DATA "BIANCHI", "680368"
1020 DATA "ACI", "06-4212"
1030 DATA "BORSA", "6292"
1040 DATA "AIUTO", "113"
9000 REM FINE AGENDA
9010 DATA "FINE AGENDA", ""
```

Per prima cosa, notiamo che i nominativi ed i numeri di telefono vanno introdotti sotto forma di stringhe contenute in linee di DATA (si può proseguire a volontà dopo la 1040), e quindi vengono salvati insieme al programma stesso.
Al RUN il programma inizia chiedendo (linea 30) il nominativo di cui cerchiamo il numero, assegnandolo alla variabile NO$.
A questo punto, la stringa NO$ viene confrontata con la stringa nulla (""): se ci si limita a premere il tasto di invio, il programma esegue un END e termina.
La linea 40 esegue un RESTORE, che riporta all'inizio la lettura dei DATA. Nel nostro caso, alla linea 1010, dove si trova il primo DATA.
Il RESTORE è necessario perché l'agenda va letta dall'inizio.
La linea 50 legge due stringhe dai primi due DATA ("BIANCHI" e "680368"), rispettivamente nelle variabili A$ e TE$.
Dopo questa operazione, il PUNTATORE dei DATA si è spostato, cioé la prossima READ leggerà dalla costante DATA successiva.
La linea 60 verifica se, a forza di leggere DATA, si è arrivati all'ultimo ("FINE AGENDA", linea 9010).
In questo caso avverte che il nome non è in agenda e ricomincia (linea 30).
La linea 9010 contiene anche una stringa nulla (""), per evitare errori nella lettura dell'ultimo TE$ alla linea 50.
La linea 70 confronta il nome cercato con il primo DATA letto; se non coincidono, torna alla linea 50, dove viene letto il prossimo.
Alla linea 80 si giunge solo se il nome coincideva: viene dunque stampato il numero di telefono.
I DATA sono due per riga solo per leggibilità: avremmo, infatti, potuto metterli anche in una sola riga. I numeri di telefono sono sotto forma di stringhe per consentire l'impiego di caratteri non numerici o zeri iniziali (vedi linea 1020).
Per quanto questo programma sia rudimentale, può venire utile ai possessori di un computer tascabile in BASIC con memoria permanente.

## CICLI

### ISTRUZIONE FOR...NEXT

Capita spesso di voler eseguire più volte un'istruzione od un gruppo di istruzioni.

Supponiamo di voler stampare i primi dieci numeri interi:

```
10 I = 1
20 PRINT I
30 I = I + 1
40 IF I<11 GOTO 20
```

Abbiamo usato I come CONTATORE, per contare cioè quante volte il programma esegue il CICLO (LOOP) costituito dalle linee 20, 30 e 40.
Alla linea 30 il contatore viene INCREMENTATO (aumentato di uno) ed alla 40 si controlla se il ciclo è stato ripetuto per il numero voluto di volte.

Tutto questo può essere ottenuto molto più semplicemente:

```
10 FOR I = 1 TO 10
20 PRINT I
30 NEXT I
```

# BASIC (Parte VI[a])

L'istruzione alla linea 10 dice:

metti I uguale a 1, esegui le istruzioni che seguono fino al NEXT I, incrementa I di 1, se non ha superato il 10 ripeti, se no continua con l'istruzione che segue il NEXT I.

Cioè il nostro programma può anche essere letto nel seguente modo:

```
10 PER (FOR) I che va da 1 FINO A (TO) 10
20 Esegui la linea 20
30 Passa al PROSSIMO (NEXT) I
```

in cui:

I si chiama VARIABILE DI CONTROLLO DEL CICLO,

1 è il VALORE INIZIALE

10 è il VALORE FINALE o DI TEST.

Al posto della linea 20 può stare qualunque altra linea, o insieme di linee, che vogliamo sia ripetuto 10 volte.

L'incremento, o PASSO, della variabile di controllo del ciclo è 1 se non specificato, ma può essere indicato esplicitamente con STEP (PASSO, appunto).

Ad esempio, vediamo un programma che calcola i quadrati dei numeri pari compresi tra 8 e 46:

```
10 FOR N = 8 TO 46 STEP 2
20 PRINT "NUMERO: ";N;" QUADRATO: ";N*N
30 NEXT N
```

Partendo da 8 ed aumentando ogni volta di 2, fino a 46, vengono stampati i numeri ed il rispettivo quadrato.

Si può anche contare all'indietro:

```
10 REM --- CONTO ALLA ROVESCIA ---
20 FOR CD = 10 TO 1 STEP -1
30 PRINT CD; " ";
40 FOR I = 1 TO 2000:NEXT I
50 NEXT CD
60 PRINT "0: LANCIATO!"
```

Questa volta abbiamo introdotto un secondo ciclo dentro il primo (linea 40), che ha solo la funzione di far passare un pò di tempo.

Si dice che ci sono due

CICLI NIDIFICATI (NESTED).

In casi come questo, bisogna ricordare che i cicli vanno chiusi (NEXT) nell'ordine inverso a quello in cui sono stati aperti (FOR).
In altre parole, l'ultimo aperto va chiuso per primo e il primo per ultimo.

Un ultimo esempio: un programma che produce la tavola pitagorica della moltiplicazione:

```
10 REM --- TABELLINA PITAGORICA ---
20 FOR I = 1 TO 10
30 FOR J = 1 TO 10
40 PRINT I * J; " ";
50 NEXT J
60 PRINT
70 NEXT I
```

Le istruzioni 30, 40 e 50 costituiscono il ciclo interno che calcola e stampa una riga della tavola, con I costante e J che varia da 1 a 10.
Notate che il PRINT alla linea 40 termina con punto e virgola (;) per non andare a capo durante la scrittura di una riga.
Per ciascuna riga si ripete lo stesso ciclo con un valore diverso di I.
Il ciclo esterno (quello in cui varia I) fa ripetere dieci volte il ciclo interno (in cui varia J).
La PRINT alla linea 60 serve per andare a capo al termine di ogni linea. Provate a far girare il programma.
Fatto?
Perché la tabellina non è venuta esattamente allineata?

Vi lasciamo trovare la risposta, con un aiuto: l'istruzione

```
45 IF I*J < 10 THEN PRINT " ";
```

mette tutto a posto.

*(continua)*

Eccoci giunti al nuovo appuntamento con il nostro corso di programmazione.

Parleremo della Programmazione o Notazione Lineare Strutturata che permette di generalizzare qualsiasi algoritmo ponendosi alla base di qualsiasi linguaggio.

## PROGRAMMAZIONE LINEARE STRUTTURATA

Trent'anni fa i Diagrammi a Blocchi (DaB) erano l'unico strumento usato per esprimere un algoritmo.

Per farlo poi eseguire da un calcolatore, si procedeva alla traduzione in un linguaggio di programmazione (Cobol o Fortran).

Vent'anni fa circa sono finalmente sorti degli strumenti alternativi alla "diagrammazione" cioè la programmazione mediante diagrammi a blocchi.

Questi strumenti hanno contribuito ad evidenziare alcuni difetti di questa forma espressiva, soprattutto sotto la spinta di una nuova generazione di linguaggi, che trovano nel Pascal il loro rappresentante di maggior successo.

I maggiori difetti riscontrati nei diagrammi di flusso o a blocchi sono:

A) Molto spesso sono illeggibili, soprattutto quando le loro dimensioni superano quelle di un semplice esercizio didattico.
La lettura andrebbe fatta un po' dall'alto al basso, un po' dal basso all'alto senza un ordine preciso.

B) Sono facilmente esposti ad errori logici : bastano pochi accavallamenti di cicli per perdere il filo del controllo, e quindi, della risoluzione del problema.

A tutto ciò si aggiungono poi i seguenti inconvenienti:

C) Scarsa praticità dovuta alla natura grafica bidimensionale poiché tutti i linguaggi di programmazione esistenti sono unidimensionali.
Sul piano concreto, tutti gli esecutori automatici comprendono un linguaggio fatto di stringhe, cioè sequenze di caratteri.
Più precisamente, tutto il programma viene visto come un'unica stringa di caratteri divisa in linee poste una di seguito all'altra.
Volendo far eseguire automaticamente un diagramma a blocchi, saremmo perciò costretti a compiere una traduzione in un linguaggio di programmazione disponibile, di natura unidimensionale, spesso modificando la struttura ordinaria.

D) Difficoltà di riconoscimento della struttura di controllo.
I blocchi possono essere disposti con la massima libertà sul foglio di carta, senza alcuna regola particolare.
Può così accadere di non riconoscere due diagrammi che differiscono solo per la dislocazione dei blocchi, ovvero ci potrebbero essere due rappresentazioni diverse dello stesso algoritmo.

Teniamo comunque sempre a mente che tutti gli inconvenienti sottolineati per i DaB, hanno come unica radice l'uso indisciplinato delle frecce di controllo.

Per ovviar e a tutti questi problemi, è stata creata la

PROGRAMMAZIONE LINEARE STRUTTURATA.

Vediamo ora alcuni degli schemi di composizione detti anche costrutti di controllo strutturati.

Ognuno di essi verrà descritto sintatticamente da alcune regole che sono sostanzialmente le stesse che si ritrovano nei linguaggi di programmazione come il Pascal o il C.

## COSTRUTTO SEQUENZA O BLOCCO BEGIN-END

Questo schema prende ordinatamente un numero arbitrario di blocchi strutturati B1, B2, ...,Bn, e ne costruisce la sequenza seguente:

```
Sintassi PLS

Begin
B1
.
B2
.
Bn-1
.
Bn
End*8
```

E' evidente che si impone di eseguire prima B1, poi B2, fino a Bn.
Se qualcuno dei blocchi si ferma, tutto il costrutto si ferma.

## COSTRUTTO IF-THEN-ELSE (se-allora-altrimenti)

Questo schema valuta inizialmente una condizione ‹cond›; se questa è vera, viene successivamente eseguito un blocco strutturato B1, altrimenti viene eseguito un blocco strutturato B2 con sintassi PLS:

```
if ‹cond› then B1 else B2
```

## COSTRUTTO WHILE (finché la condizione è vera ripeti)

Questo schema prevede una condizione ‹cond› da valutare inizialmente.

Se questa è falsa si esce immediatamente dal costrutto, altrimenti si esegue ciclicamente un blocco strutturato B, detto corpo, uscendo se e quando la condizione ‹ cond› diventerà falsa.

Sintassi PLS:

```
while ‹cond› do B
```

Vediamo ora un esempio che utilizza questi tre costrutti con un programmino in programmazione lineare strutturata che calcola il minimo comune multiplo fra due numeri naturali A e B.

```
begin
leggi A, B.
MA = A;
MB = B;
While MA ‹› MB do if MA › MB then MA = MA+A
else MB = MB+B.

MCM = MA;
scrivi MCM
end
```

E' importante notare che si esce dal ciclo while quando la condizione è falsa, nel nostro caso quando MA=MB.

## COSTRUTTO CASE (selezione a molte vie)

Abbiamo visto che il costrutto if-then-else consente di distinguere fra due possibilità corrispondenti alla verità o falsità di una certa condizione.

In pratica vi sono molti casi in cui si vorrebbe poter selezionare contemporaneamente una fra un numero più elevato di possibilità.

Pensiamo ad esempio ad un problema di questo tipo : leggere un numero fra 1 e 7 e scrivere in uscita il nome del giorno corrispondente : Lunedì, Martedì, Mercoledì, Giovedì, Venerdì, Sabato, Domenica.

Non vorremmo fare tutta la serie di confronti del tipo:

```
if I=1 then scrivi lunedì;
if I=2 then scrivi martedì:
.
.
.
if I=7 then scrivi domenica;
```

Quindi scriveremo secondo la sintassi della PLS :

```
leggi N
case N of
1: scrivi LUNEDI';
2: scrivi MARTEDI';
.
.
.
7: scrivi DOMENICA;
otherwise scrivi 'non accetto'
end-case
```

## COSTRUTTO REPEAT-UNTIL (ripeti fino a quando)

Si tratta di un tipo di ciclo in cui si ripete una sequenza di blocchi fino al verificarsi di una certa condizione.

In sintassi PLS:

```
repeat
‹B1›;
‹B2›;
.
.
.
‹BN›;
until ‹Condizione›.
```

Occorre evidenziare subito due importanti differenze fra cicli while e cicli repeat until:

nei cicli while il test di terminazione si fa all'inizio delle iterazioni e quindi l'esecutore potrebbe anche non effettuare alcun ciclo;

nei cicli repeat-until la condizione viene testata alla fine e il corpo del ciclo viene eseguito almeno una volta.

nei cicli while si esce quando la condizione è falsa mentre in quelli repeat- until quando la condizione è vera.

Supponiamo di voler scrivere un semplice algoritmo che scrive il numero di cifre decimali di cui è composto un numero N dato in ingresso:

```
begin
leggi N;
NC=0;
repeat NC=NC+1;N=N/10 until N=0;
scrivi NC;
end
```

## COSTRUTTO FOR (iterazioni enumerate)

Nella pratica programmativa, accade molto spesso di voler eseguire un numero prefissato di iterazioni.

Se disponiamo solo dei cicli while e repeat-until saremmo costretti ad introdurre delle variabili per conteggiare il numero delle iterazioni.

Se a ciò si aggiunge il fatto che questa variabile contatore viene utilizzata nello stesso ciclo (si pensi, ad esempio, alla somma dei primi N numeri) si comprende l'importanza del costrutto for che è esplicitamente destinato a questo scopo.

Purtroppo, a parte l'idea generale, non esiste un grande accordo su cosa debba fare esattamente il costrutto for.

Forniamo qui la versione che ci sembra più semplice ed opportuna in sintassi PLS:

For indice = inizio to fine do ‹B›

Occorre definire che indice è una variabile che fa le veci del contatore e che viene posta a zero all'inizio del ciclo, al valore dell'espressione inizio e incrementata dopo ogni esecuzione del corpo ‹B› del ciclo.

L'uscita è condizionata al superamento di un certo valore contenuto in una variabile speciale LIM, associata al ciclo, che viene caricata all'ingresso con il valore dell'espressione ‹fine› senza che possa subire assegnamenti nel corso dell'esecuzione dei cicli.

Supponiamo di voler scrivere la tabella dei fattoriali dei numeri da 1 a N :

```
begin
leggi N;
FATT=1;
FOR I=1 TO N do
begin
FATT=FATT*I;
scrivi "IL FATTORIALE DI",I,"E'",FATT
end;
end
```

Facciamo presente che il test di terminazione viene eseguito all'ingresso del ciclo.

Così il numero dei cicli è determinato dalla seguente regola : se ‹fine› è maggiore di ‹ inizio› allora vengono eseguiti zero cicli, altrimenti vengono eseguiti esattamente

‹fine›-‹inizio› + 1 cicli.

Naturalmente si fa l'ipotesi che la variabile indice non venga modificata nel corso dell'esecuzione dei cicli.

In effetti, tutti i linguaggi di programmazione proibiscono ogni tentativo di questo tipo.

Una buona regola di programmazione stabilisce di non fidarsi del valore dell'indice all'uscita dal ciclo a causa delle diverse interpretazioni del FOR nei i vari linguaggi.

Facciamo infine presente che molti linguaggi prevedono la possibilità di specificare un incremento della variabile indice diverso da quello unitario e anche negativo.

*(continua)*

# IL PASCAL (Parte IIª)

## ALFABETO

Il testo di un programma consiste di una sequenza finita di caratteri rapresentati visivamente da un simbolo grafico. Analogamente ai linguaggi naturali, l'insieme dei caratteri base utilizzabili nel descrivere un programma in Pascal è denominato ALFABETO. Esso include i seguenti caratteri grafici di largo uso e supportati da tutte le macchine:

Le 26 lettere dell'alfabeto inglese (minuscole e maiuscole)
a,b,c,...x,y,z
A,B,C,...X,Y,Z
Le 10 cifre arabe
0,1,2...9
I caratteri speciali + - * / = ‹ › ( ) . , : ; ' [ ] ^ " "
Il carattere "spazio" ("blank").

La composizione degli elementi di questo limitato insieme secondo le regole sintattiche definite dal linguaggio permette di costruire programmi Pascal legali. Tuttavia un qualsiasi carattere grafico non incluso nell'alfabeto può comparire all'interno di particolari sequenze riconosciute dal linguaggio (i commenti e le stringhe di caratteri), come vedremo successivamente.
Una cosiderazione finale riguarda l'uso dei caratteri "fine linea". La struttura a linee di un testo (come appare su un terminale o una stampante) è di immediata e naturale comprensione per il programmatore. Per tradurre tale struttura nella rappresentazione interna della macchina, particolari caratteri di formato, che non hanno una corrispondente rappresentazione grafica, vengono usati. Normalmente la fine della linea è rappresentata da una sequenza di uno o più caratteri di formato (ad es. "carriage return", cioè il comando di "a capo"; "line feed", over l'interlinea stabilita dal particolare sistema). Il linguaggio non definisce cosa causa la fine di una linea, ma la riconosce come elemento lessicale.

## VOCABOLARIO

Su un livello più alto di quello dei singoli caratteri, un programma è costituito da UNITA' LESSICALI o SIMBOLI BASE del linguaggio ("token"). Queste unità sono le parole del linguaggio, e il loro insieme costituisce il VOCABOLARIO del linguaggio stesso. Ogni unità lessicale è una sequenza finita di caratteri dell'alfabeto che può essere trattata come una singola entità logica. Per la facilità d'uso da parte del programmatore e la facilità di riconoscimento da parte del traduttore del linguaggio, queste unità sono chiaramente specificate. Come vedremo successivamente, le parole del linguaggio potranno essere simboli terminali e non terminali. Vediamo con l'aiuto del seguente frammento di programma quali classi di unità lessicali sono riconosciute dal Pascal.

```
.....
[controlla condizione di fine]
IF contatore = 1000
THEN stati := 'fine';
.....
```

Ogni simbolo del frammento è classificabile, senza possibilità di ambiguità, in una delle 4 classi seguenti:

identificatori e parole riservate:
"IF", "contatore", "THEN", "stato"
simboli speciali o delimitatori:
"=", ":=", "."
numeri decimali:
"1000"
stringhe di caratteri:
"fine"

In aggiunta nel testo di un programma compaiono espliciti SEPARATORI, necessari per separare elementi lessicali adiacenti: questo accade quando, senza la separazione, l'interpretazione come un singolo elemento lessicale risulta possibile.
Ad esempio, la stringa seguente (compattata dei caratteri spazio intermedi)

contatore=1000

è riconosciuta dal linguaggio come composta da tre elementi lessicali distinti, ovvero l'identificatore "contatore", il simbolo speciale "=" e il numero decimale "1000". Infatti il carattere speciale "=" non può essere usato nella composizione di un identificatore o numero. Questa specifica regola introduce degli impliciti separatori tra elementi lessicali distinti. Al contrario, la sequenza di caratteri seguente

THENstato

è interpretata come un singolo identificatore dal linguaggio, perché formata solamente da lettere. In Pascal qualunque sequenza di caratteri "spazio" e di caratteri "fine linea" è considerata un separatore. In particolare una qualunque unità lessicale non può essere spezzata su più linee di testo, una restrizione tuttavia che favorisce la pulizia e la leggibilità del programma. Inoltre anche i COMMENTI, usati solo per scopi documentativi dal programmatore e senza alcun effetto sul significato di un programma, sono considerati dei separatori.
In Pascal qualunque sequenza di caratteri delimitata dai caratteri speciali "{" e "}" (oppure "(*" e "*)" in loro alternativa) è un commento. La sequenza di caratteri precedente avrebbe la corretta interpretazione se modificata in

...THEN(* terminato)stato...

# IL PASCAL (Parte IIª)

Un commento può contenere qualsiasi carattere grafico (eccetto naturalmente "}" o la coppia "*)" perché terminatori di commento) e i caratteri di "fine linea". Quest'ultimo caso necessita di particolare attenzione da parte del programmatore. Ad esempio, nel seguente test:

```
.....
"controlla condizione di fine
IF contatore=1000
THEN stato:='fine';
"prosegui"
.....
```

la dimenticanza del carattere "}", seppure mantenendo il programma legale, ne ha radicalmente modificato il significato, avendo conglobato l'intera frase "IF" nel commento. Gli IDENTIFICATORI sono nomi usati per denotare differenti entità del linguaggio. Lessicalmente un identificatore è una sequenza di uno o più caratteri alfanumerici, di cui il primo deve essere necessariamente una lettera.
Alcuni esempi corretti di identificatori sono:

contatore opera40 z8000micro stato STATO

In particolare gli ultimi due identificatori sono considerati identici, in quanto il linguaggio non distingue le lettere minuscole dalle maiuscole.
Al contrario, le sequenze seguenti non sono riconosciute come identificatori:

13maggio AT&T alfa-beta

Le due ultime sequenze includono entrambe un carattere non alfanumerico, tuttavia il carattere "-" è presente nell'alfabeto del linguaggio ed è anche un simbolo speciale. La sequenza "alfa-beta" sarà interpretata come costituita da tre unità lessicali, ovvero due identificatori separati da un simbolo speciale. Successivamente il livello sintattico interpreterà tale frase come una operazione di sottrazione tra due oggetti denotati da identificatori. Secondo la definizione precedente un identificatore può essere di una lunghezza qualsiasi di caratteri (ovviamente limitata alla grandezza della linea), tuttavia normalmente solo i primi N caratteri sono considerati significativi da un traduttore del linguaggio. Supponendo, ad esempio, N uguale a 8, i seguenti due identificatori sono considerati identici dal traduttore

terminato terminati

Alcuni identificatori hanno un ben preciso e congelato significato nell'ambito del linguaggio.
Questi costituiscono l'insieme delle PAROLE RISERVATE o PAROLE CHIAVE del linguaggio.
Tali nomi sono riservati per particolari usi e non sono ridefinibili in altro modo dal programmatore: essi sono le chiavi che guidano nella costruzione dei costrutti sintattici del linguaggio. Il programmatore può usare questi nomi soltanto con il significato definito dal linguaggio e secondo le regole stabilite. Infatti nell'esempio precedente le parole riservate "IF" e "THEN" differiscono dagli identificatori introdotti dall'utente "contatore" e "stato", in quanto individuano uno specifico costrutto sintattico del Pascal. Il programmatore potrebbe sostituire i propri nomi con degli altri senza alterare la validità sintattica della frase, al contrario questo non è vero per le parole riservate. La seguente frase non è infatti legale in Pascal:

```
THEN contatore=1000
IF stato:='fine';
```

La lista delle parole riservate del Pascale è la seguente:

| | | | | |
|---|---|---|---|---|
| AND | ARRAY | BEGIN | CASE | CONST |
| DIV | DO | DOWNTO | ELSE | END |
| FILE | FOR | FUNCTION | GOTO | IF |
| IN | LABEL | MOD | NIT | NOT |
| OF | OR | PACKED | PROCEDURE | PROGRAM |
| RECORD | REPEAT | SET | THEN | TO |
| TYPE | UNTIL | VAR | WHILE | WITH |

Tutte le parole riservate compaiono come simboli terminali nelle carte sintattiche del linguaggio.

Accanto agli identificatori definiti dal programmatore ed a quelli riservati, esiste inoltre un insieme di IDENTIFICATORI STANDARD con un significato predefinito dal linguaggio.
Essi denotano una varietà di entità diverse del linguaggio e globalmente costituiscono l'ambiente iniziale standard di un programma Pascal.
Ad esempio, le seguenti comuni funzioni matematiche sono direttamente fornite dal linguaggio e denotate da nomi standard:

| | | | |
|---|---|---|---|
| abs | sin | cos | sqrt |

A differenza delle parole riservate, tuttavia tali identificatori non sono congelati nel significato, ma possono essere ridefiniti dal programmatore per denotare differenti entità.
Di seguito è riportata la lista completa degli identificatori predefiniti riconosciuti dal Pascal:

| | | | | | | |
|---|---|---|---|---|---|---|
| abs | arctan | boolean | char | chr | cos | dispose |
| eof | eoln | exp | false | get | input | integer |
| ln | maxint | new | odd | ord | output | pack |
| page | pred | put | read | readln | real | reset |
| rewrite | round | sin | sqr | sqrt | succ | text |
| true | trunc | unpack | write | writeln | | |

# IL PASCAL (Parte IIª)

I SIMBOLI SPECIALI sono o semplici caratteri speciali o coppie adiacenti di essi, tuttavia interpretate come singole entità logiche.
Essi denotano tipicamente le differenti operazioni fornite dal linguaggio e in generale tutta la punteggiatura richiesta dalla sintassi. La scelta di noti caratteri speciali per esplicitare tali funzioni è dettata ovviamente dalla vicinanza simbolica con le familiari notazioni della matematica e grammatica.
Appare molto più immediato ed intuitivo infatti descrivere le operazioni di somma e sottrazione con i caratteri speciali

```
"+" e "-"
```

piuttosto che con identificatori

```
"PLUS" e "MINUS"
```

o altri simboli.
La seguente frase Pascal è evidentemente di immediata lettura:

```
a + b - c
```

Essendo caratteri speciali, tali simboli sono distinguibili senza l'ausilio di separatori quando compaiono tra identificatori o numeri.
Ad esempio.

```
contatore=1000
```

è interpretabile correttamente anche in mancanza di spazi intermedi tra le tre parole.
Al contrario, i separatori possono alterare l'interpretazione di simboli speciali costituiti da coppie di caratteri.
Ad esempio, nella frase

```
stato : = 'fine';

il simbolo speciale

":="
```

non è infatti in tal caso riconosciuto; invece i due semplici simboli speciali

```
":" e "="
```

sono identificati per la presenza di uno spazio separatore tra essi.
L'insieme dei simboli speciali riconosciuti dal Pascal è il seguente:

```
+ - * / ‹ › = ‹= ›= ‹› := , ;
: ' . .. ^ ( ) [ ] (. .) " "
(* *)
```

I NUMERI trattati dal linguaggio sono di due tipi, interi o reali. Un numero intero è una sequenza di cifre eventualmente preceduta da un segno.
Un numero reale ha una parte decimale e/o un fattore di scala. Entrambi denotano i valori costanti di due tipi predefiniti nel linguaggio, rispettivamente il tipo

```
INTEGER e REAL.
```

Seguendo la sintassi, alcuni esempi corretti di numeri interi sono:

```
5946      +85      -0      5      023
```

Analogamente alcuni numeri reali legali sono:

```
3.52E+7      -1.57E9      +0.0      1.1E5
```

La scritta

```
3.52E+7
```

è una forma per indicare

```
3.52 x 10^7.
```

Non hanno invece una forma corretta le seguenti sequenze:

```
.7      5.E2      -1.4E
```

Una STRINGA DI CARATTERI è costituita da una sequenza qualsiasi di caratteri grafici racchiusa dal carattere speciale "'".
Il valore denotato dalla stringa è dato da tutti i caratteri grafici inclusi eccettuati i due caratteri delimitatori. In particolare, se si vuole che il carattere "'" stesso compaia come valore nella stringa, allora deve essere rappresentato da una coppia adiacente di "''".
Alcuni esempi corretti di stringhe di caratteri sono:

```
'12345' 'a'      '#$%&'      'l''ape' ''
```

Naturalmente i caratteri spazio e i commenti all'interno di stringhe perdono il significato di separatori e sono considerati come normali caratteri.
I caratteri "fine linea" invece non sono permessi all'interno di stringhe di caratteri. La lunghezza di una stringa non può quindi superare la dimensione fisica di una linea.
Come vedremo successivamente le stringhe denotano i valori costanti del tipo predefinito CHAR (se la lunghezza è 1) o di un tipo impaccato array (se la lunghezza è maggiore di 1).

*(continua)*

# DENTRO L'MSX (Parte IIIª)

## IL BASIC MSX

Il BASIC MSX è sostanzialmente il BASIC Microsoft standard versione 4.5 con numerose estensioni nell'area della grafica e della produzione di suoni.
Le principali mancanze sono invece le Procedures, le strutture WHILE/WEND e REPEAT/UNTIL e la completa abbreviazione delle parole chiave.
Viene reso disponibile un editor a schermo intero che permette di correggere una qualsiasi linea dello schermo. La correzione effettiva della linea avviene quando viene battuto il tasto [RETURN] mentre il cursore si trova su qualsiasi carattere della linea in questione. Nella fase di introduzione non viene effettuato alcun controllo di correttezza.
Una linea può essere composta al massimo da 255 caratteri e può contenere più di una istruzione.
Più istruzioni che si trovino sula stessa linea devono essere separate dai due punti. Un'istruzione REM fa sì che l'esecuzione prosegua dalla linea successiva. I numeri di linea devono essere compresi nell'intervallo 0-65529, compresi gli estremi.
Gli spazi non sono necessari e vengono ignorati dall'interprete (tranne quando fanno parte di una stringa). Una conseguenza di questo fatto è che le parole chiave non possono essere parte di nomi di variabili (dei quali sono riconosciuti solo i primi due caratteri).
Vengono considerate diverse le lettere maiuscole e quelle minuscole. Le parole chiave possono essere scritte sia maiuscole che minuscole.
A fianco della tastiera principale vi sono due gruppi di tasti. Sulla destra il gruppo dei tasti per l'editing - i tasti con le frecce che indicano la direzione dello spostamenteo del cursore e le opzioni di inserimento e cancellazione.
Sulla sinistra c'è un insieme di cinque tasti funzionali. Al momento dell'accensione questi tasti sono predefiniti con dieci funzioni.
Quelle corrispondenti ai tasti funzione "minuscoli" (cioè senza che sia premuto il tasto [SHIFT]) sono mostrate sull'ultima linea dello schermo. Questa visualizzazione può essere evitata, pur mantenedo la definizione delle funzioni, tramite il comando KEY OFF. Per ridefinire uno qualsiasi dei tasti funzione, si usa l'istruzione KEY x,"stringa".
Per esempio, si può definire nel seguente modo il tasto funzione 1 in modo che serva per mandare in esecuzione un programma.

```
KEY 1,"RUN"+CHR$(13)
```

La visualizzazione di una lista di programma o un'esecuzione può essere interrotta premendo il tasto [STOP] una volta. Premendolo nuovamente viene riattivata l'azione precedentemente interrotta.
Per terminare definitivamente l'operazione in corso si devono premere contemporaneamente lo [STOP] e il tasto [CTRL] (CONTROL).

Per coloro che non hanno familiarità con il BASIC Microsoft è opportuno porre in evidenza una serie di caratteristiche:

1 – La messa a punto dei programmi è semplificata dall'uso delle istruzioni TRON e TROFF. Se viene eseguito il comando TRON, direttamente o all'interno di un programma, questo fa sì che venga visualizzato il numero di ogni linea che viene eseguita; solitamente eventuali altre visualizzazioni sono sovrascritte.

2 – Blocchi di linee possono essere cancellati con l'istruzione DELETE X-Y, mentre le linee di un programma possono essere rinumerate tramite il comando RENUM X,Y.
Questa possibilità si rivela utile nell'identificare eventuali salti a numeri di linea non esistenti. In tal caso infatti il BASIC MSX non segnala un errore; viene invece eseguita la linea con numero immediatamente superiore a quello indicato nel salto.

3 – Tutte le variabili che iniziano con un particolare carattere, o con un insieme di caratteri, possono essere dichiarate di un particolare tipo: DEFINT per variabili intere, DEFSNG per variabili in singola precisione, DEFDBL in doppia precisione e DEGSTR per le stringhe.

4 – I vettori pososno essere cancellati per mezzo del comando ERASE.

5 – Al momento dell'accensione lo spazio allocato per la memorizzazione delle stringhe è di 200 byte. Un'ulteriore allocazione necessita dell'uso di un comando di CLEAR. Ad esempio, per riservare 1000 byte si usa CLEAR 1000.

6 – La funzione FRE(0) ritorna l'ammontare della memoria disponibile per la memorizzazione del programma.

7 – La funzione per le stringhe MID$ può essere usata per sostituire uno o più caratteri in un'espressione.

8 – La struttura GOSUB/RETURN consente di specificare un indirizzo di ritorno: RETURN "numero di linea".

# DENTRO L'MSX (Parte III[a])

## VARIABILI E FUNZIONI

I nomi delle variabili possono essere di qualsiasi lunghezza, ma devono iniziare con una lettera. Solo i primi due caratteri sono significativi. Le lettere minuscole vengono distinte da quelle maiuscole.
Non è necessario inizializzare le variabili: viene assunto il valore zero. Le variabili di tipo vettore non devono necessariamente essere dimensionate, tranne quando il numero di elementi supera undici: quindi S(0) e S(10), per esempio, non hanno bisogno di un esplicito dimensionamento.
Una variabile che non è dichiarata di un tipo particolare si assume che sia reale in doppia precisione, e viene memorizzata con quattordici cifre significative. Le variabili dichiarate in singola precisione sono memorizzate con una massima precisione di sei cifre. Gli interi sono compresi tra -32768 e 32767.
Una variabile in singola precisione viene distinta tramite un punto esclamativo alla fine, come SP!, mentre un'intera viene contraddistinta da un segno di percentuale (%). Se è necessario dichiarare una variabile in doppia precisione, viene usato il simbolo #.
Come accennato in precedenza, le varianti dell'istruzione

```
DEF
```

possono essere utilizzate per definire globalmente i tipi delle variabili.
Se ad esempio viene usata l'istruzione

```
DEFINT A
```

allora tutte le variabili che iniziano con A verranno trattate come variabili intere. Inoltre non verrà fatta alcuna distinzione tra 'A' e 'A%'.
Il dimensionamento delle variabili è molto semplice: si consideri a questo proposito l'istruzione

```
DIM X(20,40),Y(80,160)
```

che definisce due matrici rettangolari; la prima ha 20 righe e 40 colonne mentre la seconda ne ha 80 per 160. Esiste il limite di 255 sul numero delle dimensioni, ma il numero degli elementi è limitato solamente dalla disponibilità della memoria.
Poiché una qualsiasi variabile vettore che abbia meno di 11 elementi non deve essere esplicitamente dimensionata, ne consegue che, per un utilizzo ottimale dello spazio di memoria, è opportuno dimensionare quelle che hanno un numero di elementi minore di 11.

Il BASIC MSX comprende un'istruzione

```
SWAP X,Y
```

che serve per scambiare il contenuto delle variabili X e Y. Le variabili in questione devono però essere del medesimo tipo. Un'altra istruzione in relazione con le variabili è la

```
VARPTR X
```

che ritrona la locazione di memoria dell'oggetto specificato - se quest'ultimo è stato dichiarato.
Le variabili di sistema dell'MSX sono:

`TIME` – E', in un certo senso, l'orologio del sistema. Viene incrementato ogni 1/50 di secondo.

`BASE(N)` – Ritorna la locazione della tabella specificata della RAM video, indipendentemente dalla modalità di visualizzazione.

`VDP(N)` – Ritorna il valore del registro a sola scrittura specificato dal VDP, o viceversa il registro di stato di quest'ultimo.

`SPRITE$(PATTERN#)` – Questa variabile viene usata per definire ognuna delle 256 possibili 8*8 forme di sprite, oppure una delle 64 forme 16*16. Contiene una stringa di 32 caratteri. Il codice di ciascuno di questi caratteri determina la configurazione dei bit in un byte della definizione dello sprite.

## FUNZIONI

Sono disponibili tute le funzioni che siamo abituati a trovare nel BASIC.
In particolare, se X è il nome di una arbitraria:

```
ASC(X$), CHR$(X), LEN(X$), INT(X), EXP(X), ABS(X), LOG(X), HEX$(X), OCT$(X), BIN$(X), SGN(X), RND(X), MID$(X$,N,n), RIGHT$(X$,N), LEFT$(X$,N), STR$(X), VAL(X$), STRING$(N,X$), INSTR(X$,Y$), ATN(X), COS(X), SIN(X), SQR(X), TAN(X), TAB(n), SPC(n), INKEY$, INPUT$(n), PEEK(X), POKE X,N, FRE(0), FRE"", USR X(n), VARPTR(X), VARPTR(#X), CINT(X), POS(n) e LPOS(n).
```

Le estensioni comprendono:

CDBL(X) e CSGN(X)
che convertono X rispettivamente a doppia e singo-

# DENTRO L'MSX (Parte III[a])

la precisione.

VPEEK(n) e VPOKE n,X
sono le istruzioni equivalenti, per la RAM video, di PEEK e POKE.

STICK(n) e STRIG(n)
ritornano la direzione e lo stato di trigger del joystick (oppure del cursore e della barra di spaziatura).

POINT(x,y)
ritorna il colore del pixel che si trova alle coordinate X e Y.

PLAY(canale)
indica lo stato di una o di tutte le code musicali.

EOF(file#)
ritorna -1 se è stata raggiunta la fine di un file sequenziale, alltrimenti 0.

PAD(n):
in questo caso n determina quale parametro dello statro di un paddel a controllo tattile viene ritornato.

PDL(paddel#)
ritorna il valore di un paddle.

## ISTRUZIONI PER LA GRAFICA

L'intera gamma dei 15 colori più il trasparente è disponibile in tutte le modalità di visualizzazione.
Le istruzioni per la grafica possono essere divise in tre categorie:

1 - Formattazione generale e colore
2 - Sprite
3 - Alta risoluzione

## COMANDI DI USO GENERALE

La modalità di visualizzazione sullo schermo viene impostata con l'istruzione SCREEN:

SCREEN 0 modalità testo 40*24
SCREEN 1 modalità testo 32*24
SCREEN 2 modalità grafica ad alta risoluzione (HRG)
SCREEN 3 modalità multicolore

Questi comandi possono essere utilizzati anche per impostare la dimensione degli sprite, la velocità di trasferimento dati su cassetta e le opzioni per la stampante. Quando vengono usate per selezionare una modalità testo, l'insieme dei caratteri viene copiato dalla ROM alla VRAM.
L'istruzione SCREEN non deve essere usata qualora l'utente abbia la necessità di fare riferimento ad un qualsiasi carattere ridefinito.

### MODALITA' TESTO

La larghezza effettiva dello schermo sia per la modalità a 40 colonne sia per quella a 32, può essere diminuita con l'istruzione

WIDTH

(ad esempio WIDTH 20 per avere lo schermo su 20 colonne soltanto).
Si noti che in entrambe le modalità testo la colonna più a destra e quella più a sinistra non vengono usate. La maggior parte dei ricevitori TV non visualizzano la colonna più a sinistra.
In entrambe le modalità i caratteri possono essere messi in una colonna particolare scrivendo sulla tabella dei nomi della RAM video. Ad esempio, per mettere una A maiuscola nella prima riga delle due colonne più a sinistra nella modalità a 40 colonne, si usa la

VPOKE

delle locazioni 0 e 1 con il valore 65.
I colori globali per una qualsiasi modalità di visualizzazione vengono impostati tramite l'istruzione

COLOR.

Il formato della sua sintassi è il seguente:

COLOR primo-piano, sfondo, bordo

L'impostazione di default è 15,4,4. Nel modo a 40 colonne il colore del bordo non può essere specificato separatamente e assume quello dello sfondo.
Nei modi schermo 1, 2 e 3 si possono ottenere uno sfondo ed un bordo color grigio inchiostro con l'istruzione COLOR 14,1,1.
Si noti che nelle modalità grafiche il nuovo colore dello sfondo diventa effettivo solo dopo un'istruzione CLS.
Nella modalità testo a 40 colonne, il VDP non consente di usare altro colore.
Non è possibile usare gli sprite. Viceversa nel modo a 32 colonne il VDP visualizza il testo in un colore differente da quelli scelti per mezzo dell'istruzione COLOR. Però a questo scopo non esiste comando BASIC. Quindi per ottenere sullo schermo più di un colore di primo piano o di sfondo, è necessario apportare dei cambiamenti alla tabella dei colori della VRAM. Per far ciò si usa una VPOKE.
Un'operazione del tutto simile deve essere effettuata per ridefinire dei caratteri.

*(Continua)*

# SPECIALE SIMULAZIONE

Fin da quando comparvero i primi Home Computer i simulatori di volo sono sempre stati uno dei generi preferiti e maggiormente sfruttati dalle case produttrici di programmi. Indipendentemente dal fatto che li apprezziate o meno (anche se è un dato di fatto che la maggior parte non ci gioca per l'eccessivo impegno necessario ad imparare ad usare i numerosi comandi che nella maggior parte dei casi sono tutt'altro che agevoli) rappresentano indubbiamente uno degli sviluppi più interessanti intrapreso dal software applicativo. Con l'avvento dei 16-Bit poi si è finalmente aggiunto, grazie alle ampliatissime possibilità grafiche, quello straordinario realismo che fa di alcuni programmi dei veri e propri capolavori.
Partendo dal fatto che un simulatore di volo è prima di tutto, per la stessa definizione, un simultaore, risulta fin da subito evidente come il suo obiettivo principale sia quello di riprodurre quanto più fedelmente possibile la realtà in tutti i suoi più svariati aspetti. Tornando agli albori del software, quegli ormai dimenticati titoli che allora erano tanto acclamati non potrebbero non farci sorridere nella loro rudimentale essenzialità; anche se nonostante tutto, con un pò di fantasia, si poteva ugualmente calarsi nei panni dell'intrepido pilota impegnato in evoluzioni mozzafiato. Ad ogni modo uno dei problemi principali per quei così detti simulatori della prima generazione (grafica a parte) era una totale mancanza nella maggior parte di un vero e proprio obiettivo conforme ad una specifica missione. Si finiva così per vagabondare per il cielo cercando di schiantarsi il più tardi possibile. Una svolta determinate venne probabilmente con il celeberrimo GUNSHIP della Microprose (anche se altri programmi come SOLO FLIGHT e SUPER HUEY avevano già imboccato la strada buona) che accompagnava una più dignitosa simulazione di elicottero con una complessa quanto altamente coinvolgente struttura di supporto che prevedeva la scelta degli armamenti, delle missioni, la possibilità di fare carriera con il proprio pilota, aspetto che fu forse la ciliegina sulla torta e che permise al programma di conseguire onorificienze e riconoscimenti un pò da tutta la stampa specializzata.
La svolta successiva che ci ha permesso di arrivare a quelli della terza generazione è stato indubbiamente l'avvento dei computer a 16-bit che grazie alle loro caratteristiche tecniche di gran lunga superiori hanno consentito di ottenere un realismo che fino a qualche anno fa poteva sembrare irrangiungibile. Il primo programma che sconvolse tutti gli utenti con una solida e oltremodo fluida grafica vettoriale in tre dimensioni e con la possibilità di svariare il punto di vista dell'utente fu l'ultrafamoso F/A-18 INTERCEPTOR dell'Electronic Arts. Per la prima volta il panorama non era più limitato quasi solamente a due zone di diverso colore rappresentanti presumibilmente il cielo e la terra e alcune indecifrabili figure triangolari, ma bensì un'abbondanza di dettagli e piccoli tocchi che aumentavano notevolmente quella strana sensazione "d'esserci veramente". In ogni caso INTERCEPTOR fu anche criticato per la sua irrealisticità nei movimenti, anche se noi crediamo che questo abbia facilitato una così improvvisa diffusione aprendo la strada a tutti i seguenti programmi sempre più evoluti come F15 STRIKE EAGLE II della Microprose, FIGHTER BOMBER dell'Activision, F29 RETALIATOR dell'Ocean, LHX ATTACK CHOPPER dell'Electronics Arts, F16 FALCON della Spectrum Holobyte, F16 COMBAT PILOT della Digital Integration e via dicendo... Ognuno di questi simulatori della nuova generazione è caratterizzato da un numero sempre maggiore di opzioni che esulano da quella che è la fase di volo vera e propria. Il "background" del gioco si è fatto via via più esteso consentendo al nuovo cadetto di vivere un'esperienza quanto più completa e attinente alla realtà. E così ecco comparire uffici per l'arruolamento, accuratissime sezioni d'allenamento, varietà di missioni da affrontare, possibilità di determinare autonomamente l'armamento da adottare e soprattutto (come già succedeva in GUNSHIP) doversi preoccupare del proprio "curriculum interattivo" che si aggiorna progressivamente procedendo nelle missioni. Aspetto questo che è stato poi ulteriormente implementato anche per quanto riguarda lo scenario di ogni missione che subisce le conseguenze delle vostre azioni precedenti e così se ad esempio avrete già distrutto tutte le postazioni antiaereo (SAM) la volta dopo avrete un problema in meno a cui fare attenzione, per non parlare poi del campo di battaglia di F29 che va avanti da solo autonomamente lasciandovi l'iniziativa di intervenire a vostra discrezione nel caso non si tratti di un bersaglio primario.
Verrebbe a questo punto da porsi una domanda, ma che cosa caratterizza un simulatore dell'ultima generazione? Probabilmente fino ad oggi il più completo e tecnicamente accurato deve essere considerato FLIGHT SIMULATOR 4 (FS4) della Sublogic/Microsoft che non solo presenta una fedele e oltremodo veritiera simulazione di un Cessna 182, ma anche degli scenari dinamici con tanto di strade trafficate e minuziosissimi particolari come ad esempio degli yachts per aggiungere un ulteriore tocco di realismo. E in più c'è perfino un costruction-set che permette di modificare molti dei parametri pre-impostati. Analizziamo comunque le caratteristiche comuni di tutti i simulatori citati in precedenza per vedere dove questi ancora pecchino di verismo. Dunque ognuno vi mette al comando di un aereo, oppure una lista fra cui scegliere ognuno dei quali viene presentato con le sue specifiche caratteristiche relative alle dimensioni, velocità, autonomia, punto di stallo, resistenza alla forza gravitazio-

nale, inclinazione massima di salita etc.. Ci sono poi i vari scenari e le numerose missioni con i loro particolari obiettivi da portare a termine vedendosela direttamente con le forze nemiche che vi impegneranno in uno scontro diretto, il problema dei rifornimenti e della strategia vincente da adottare. Ma tutte queste caratteristiche, secondo il parere di un vero pilota militare, per quanto sicuramente indispensabili, non sono sufficienti per assicurare quel tanto agognato realismo che ogni simulazione si prepone come meta finale. Quello che caratterizza infatti una vera missione odierna e che può influenzarne l'esito anche al di là dell'abilità dei piloti o delle caratteristiche tecniche dei velivoli è infatti l'affiatamento che si riesce a formare all'interno di una pattuglia che determina la sincronia con la quale vengono messe in atto le varie manovre fondamentali per la riuscita dell'azione. Se ora andiamo a rivedere tutti i simulatori sopracitati possiamo facilmente verificare come la maggior parte di questi proponga un solo aereo in missione contro moltitudini di agguerritissimi nemici, cosa che rende piuttosto evidente una componente irrealistica ancora lungi dall'essere completamente annullata.
Fortunatamente però ci si sta già muovendo anche in questa direzione come testimoniano alcuni programmi in circolazione che cercano di ovviare a questo inconveniente in maniera piuttosto efficace. Ci riferiamo in particolare sia ai due simulatori della Lucasfilm BATTLEHAWKS/THEIR FINEST HOUR sia ad A10 TANK KILLER della Dynamix che si possono considerare per gli accorgimenti che vado ad esporvi ancora un passo in avanti rispetto agli altri. Entrambi i simulatori consentono o necessitano di essere affrontati tenendo bene in considerazione per la riuscita della missione che si sta effettuando un'operazione di gruppo con più unità partecipanti. In BATTLEHAWKS ad esempio vi sarà certo capitato di sentirvi ormai perduti con un aereo nemico proprio dietro di voi pronto a colpirvi quando è intervenuto provvidenzialmente un vostro compagno a salvarvi da un'ormai disperata situazione. In A10 invece viene adottato l'interessantissimo collegamento via radio che vi informa prontamente se state andando fuori posizione o se un vostro compagno è sotto il fuoco nemico incaricandovi di andare a dargli una mano o magari di dividervi per un'azione differenziata.
Ma oltre a questo THEIR FINEST HOUR presenta anche un nuovo accorgimento che lo caratterizza ulteriormente e che crediamo diverrà un punto fisso per i prossimi simulatori in fase di sviluppo, mi riferisco all'intelligenza simulata dei piloti da voi non direttamente controllati che evolvono individualmente le loro caratteristiche personali permettendovi di trasferirli a vostro piacimento da una base ad un'altra nel tentativo di allestire una squadriglia quanto più idonea possibile per affrontare una determinata missione.

## QUALE COMPUTER?

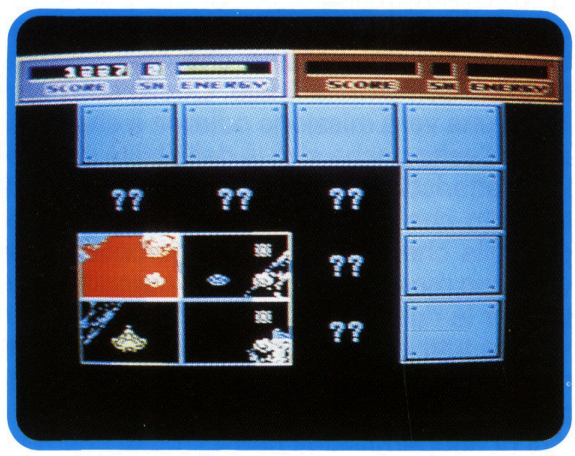
Quale computer potrebbe rappresentare la miglior base per l'utilizzo di un simulatore di volo? Senza dubbio la scelta più indicata case sull'MS-DOS modello AT (IBM o qualsiasi PC compatibile) piuttosto che sul più lento modello XT, che si rivelerebbe poco adatto ad elaborare la grafica e i dati di un flight simulator qualsiasi. Tanto per cominciare, molte delle migliori simulazioni di volo vengono sviluppate per PC MS-DOS compatibili. Inoltre, la versione per PC di Flight Simulator della SubLogic è finora l'unica disponiblie aggiornata: ha infatti raggiunto la quarta versione mentre quelle per altri personal non hanno superato la seconda, addirittura la prima per l'MSX.
Molti simulatori di volo sfruttano bene le capacità grafiche dell'ottima scheda VGA del PC (256 colori su schermo contemporaneamente) così come l'opzione per il joystick analogico e altri accessori simili per consentire movimenti più precisi se comparati a quelli prodotti dagli input digitali degli usuali joystick. Così come i PC più veloci (286, 386 e ora 486) producono animazioni molto più fluide nel volo, si è pensato di sfruttare l'hardware specifico che costituisce oggi la parte più potente dei nuovi PC. FLIGHT SIMULATOR 4 usa le istruzioni specifiche del 286 e del 386 e FALCON 3.0 della Spectrum Holobyte sarà in grado di sfruttare il coprocessore matematico se presente. E la miglioria apportata al secondo simulatore citato gli dona un incredibile realismo aereodinamico. Il sonoro sta subendo anch'esso un certo sviluppo grazie alla sempre maggiore diffusione di schede sonore. La maggior parte degli sviluppatori utilizza l'ADLIB mentre altre schede - GAMES BLASTER, ROLAND LAPC-1 - stanno assumendo sempre maggiore importanza.
E proseguendo coi vantaggi, c'è da dire che sono stati creati una serie di 'optional' progettati tenendo presente il PC come piattaforma Hardware, ma possono essere facilmente adattati per l'uso con altre macchine. Il MAXX è un joystick specifico 'tipo cloche' molto adatto per le simulazioni di grossi velivoli o di aereoplani civili. La ADVANCED GRAVIS offre un joystick molto più tradizionale che dà una maggiore impr ssione di ambiente da combattimento aereo.
Tuttavia, la Su _ogic possiede la serie definitiva di controlli di volo almeno per ora. Chiamato FLIGHT CONTROLS 1, il marchingegno comprende una cloche, acceleratore ed altri interruttori come un selettore di flap il tutto collegato ad un ampio pannello di controllo. E' disponibile persino un set di pedali di controllo.
Il punto debole del PC? Uno solo, il prezzo. Un PC 386 dotato di scheda VGA, monitor a colori e hard disk abbastanza accessibile costa comunque più di tre milioni, vedi l'ultimissimo Amstrad. Sebbene i modelli 286 con VGA stanno diventando più accessibili stiamo intorno al milione e mezzo come minimo - non è ancora una fascia di prezzo adatta alla maggior parte degli appassionati di simulazione. A questo punto, dobbiamo solo sperare che qualche casa di buona volontà produttrice di hardware produca un PC AT con VGA, presa joystick di serie a scheda sonora che non superi - nel suo prezzo globale - il milione. Continuiamo a sperare...

# RECENSIONE SOFTWARE


## 3D POOL

Prodotto da FIREBIRD
Potete trovarlo da:
Msx Club Italia - Cp 34
20075 Lodi Centro (MI)
Disponibile sia su disco che su cassetta
Prezzo: 7.000 lire

Come si poteva restituire la giovinezza a un tipo di gioco già ben sviluppato su microcomputer? I biliardi automatici avevano raggiunto il top della qualità e della ricchezza. La novità di questa edizione è lo sviluppo del gioco in 3 Dimensioni, cosicché gli ideatori di soft rilanciano la moda del "pool". Dopo il superbo Billiard Simulator su ST, 3D Pool vi offre uno scenario e un gioco molto semplici, ma comunque soddisfacenti per gli appassionati del genere. Unico punto interessante della partita, il giocatore non deve soltanto orientare la stecca, ma anche girare introno al tavolo per scegliere il suo angolo di tiro. Maneggiato con il joystick, l'orientamento 3D della scena è molto fluido. Ma, per contro, tutti gli elementi del gioco sono semplificati all'estremo, il che non accadeva per Billiard Simulator. La grafica non è eccellente e i menù di selezione si riducono a una fredda pagina di testo, ma bisogna considerare che dovendo girare anche sugli Msx 1 il lavoro fatto dai programmatori è stato notevole. Niente fronzoli dunque, solo un gioco efficace da vedere. Riservato agli appassionati. Sono previsti adattamenti su tutti i microcomputer.




## BLASTEROIDS

Prodotto da IMAGE WORKS
Lo trovate da:
Msx Club Italia - Cp 34
20075 Lodi Centro (MI)
Disponibile solo su disco
Prezzo: 7.000 lire

E' dentro alla vecchia terraccotta che vengono preparate le migliori zuppe.

Vi chiederete, è forse ora di pranzo che si parla di pentole e zupper?

No! Non è che un modo un po' insolito per riparlarvi, diciamo così, di una vecchia "zuppa".

Riprendendo il tema di Asteroid (il precursore dei giochi d'azione spaziale), Blasteroids ripropone in modo semplice questo tema di gioco.

Dovete esplorare diversi settori della galassia pieni di meteoriti e vascelli extraterrestri.

Motivo di interesse è dato dal fatto che la distruzione dei vascelli alieni vi permette di migliorare l'armamento e la potenza di propulsione del vostro vascello.

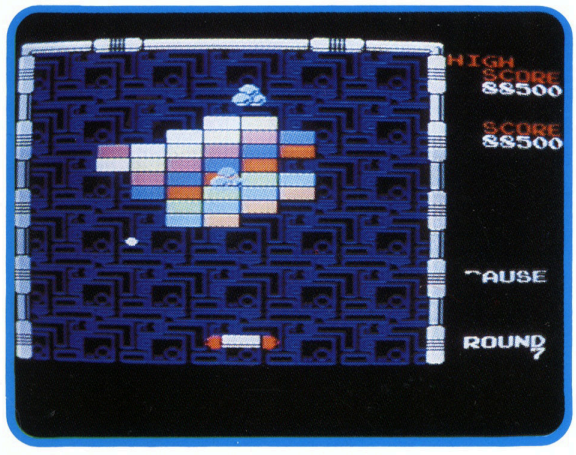
Il vostro ultimo obiettivo è annientare Mukor (una creatura immonda sorgente di tutti i vostri fastidi), forte dell'esperienza e delle armi acquisite durante le fasi precedenti.

# RECENSIONE SOFTWARE

Gli effetti grafici sono eccellenti, ma anche le animazioni multiple che prevedono un numero considerevole di oggetti nello schermo.

Pur avendo visto anche la versione Amiga, dobbiamo dire che su MSX (funziona anche con l'Msx 1) la grafica è buona anche se non al livello di giochi per l'Msx 2 o la stessa versione Amiga.

Il gioco funziona sotto MSX-DOS e quindi esiste solo su disco.




**ARKANOID**
Prodotto da TAITO
Lo trovate da: Msx Club Italia - Cp 34
20075 Lodi Centro (MI)
Disponibile sia su disco che su cassetta
Prezzo: 7000 lire

Lo scenario di Arkanoid è assai stupido e privo di interesse.

Esso si articola, in breve, nella storia di un vascello spaziale attaccato nello spazio dai soliti misteriosi alieni, da cui si distacca una capsula di salvataggio che va a finire in un complesso Dho di energia pura, e così via.

La sola cosa da mettere in rilievo in questo game (a parte il fatto che resta sempre un buonissimo test per i vostri riflessi), è che questa è la versione ufficiale Msx della versione arcade e che la conversione è decisamente ottica, addirittura migliore della versione per il Pc Ibm.

L'animazione è impeccabile, e la gestione del sonoro superba.

Partendo da uno spaccamattoni che ricorda il primo mitico Breakout, la Taito è riuscita a farne un Hit dove i differenti bonus a disposizione del giocatore rendono difficile ma non irraggiungibile l'obiettivo finale.

# SPLENDIDI INEDITI DEI MOSTRI SACRI DEL JAZZ

- ○ 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

7 COMPACT DISC AL PREZZO DI L. 84.000


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