PLAY ON TAPE

COMPUTER

manuale 6

Per i possessori di computer VIC 20 e COMMODORE 16

LINGUAGGIO MACCHINA: 1° LEZIONE

PSEUDOCODICE: I numeri a caso

La smazzata

IL BASIC: 3ª lezione

EV EDITORIALE VIDEO

PLAY ON TAPE Computer numero 6. Il manuale è sempre più ricco, sempre più indispensabile per i possessori di VIC 20 e, a partire da questo momento, anche per i possessori di COMMODORE 16, la nuova favolosa macchina che si sta procurando molti appassionati.

Per i nuovi lettori di PLAY ON TAPE Computer c'è la possibilità di richiedere i numeri arretrati che, per quanto riguarda la maggior parte di articoli e lezioni (ad esempio Pseudocodice e corso di Basic) sono validi anche per gli utilizzatori di COMMODORE 16.

In questo numero c'è la sesta lezione di Pseudocodice che presenta un'esercitazione pratica molto interessante per coloro che si sono accostati alla programmazione: la distribuzione di un mazzo di 52 carte a quattro giocatori. La cosiddetta "smazzata" è davvero indispensabile nel bagaglio pratico di un programmatore.

E arriviamo subito al piatto forte di questo numero: la prima lezione di LINGUAGGIO MACCHINA. L'autore ha voluto porre in rilievo un concetto che spesso viene preso sotto gamba: per diventare esperti di L/M è utile e necessario puntualizzare i concetti di elettronica digitale, ed è giusto questo l'argomento della prima puntata.

Prosegue frattanto il corso di BASIC, valido sia per il VIC 20 che per il COMMODORE 16, che in questo numero si occupa di parecchie funzioni principali che certo sono già familiari alla maggior parte dei lettori. Tuttavia è utile inquadrarle in un discorso scientifico che integra perfettamente la pratica.

A partire da questo mese la cassetta contiene 5 programmi validi per il VIC 20 e altrettanti per il COMMODORE 16. Per ottenere questo risultato, senza minimamente sacrificare gli appassionati di ciascuno dei due computer, abbiamo notevolmente aumentato la lunghezza del nastro.

Il lato VIC 20 si presenta così addirittura molto più ricco del solito con un programma di Adventure, "Redknight", che si snoda in altri tre favolosi programmi di gioco; poi c'è "Oroscopo", "Risparmiatore", "Api e Ragni" e il gioco del "Quindici" tridimensionale.

Il lato C. 16 presenta un programma di pronostici per il "Totocalcio", il favoloso gioco di strategia "Wargame", un programma per stabilire il proprio "Bioritmo" e infine due giochi molto divertenti "Colour search" e la "Roulette".

Ripetiamo come fare per ricevere a domicilio i numeri arretrati: bisogna inviare la richiesta su vaglia intestato a Editoriale VIDEO - via Castelvetro 9 - 20154 Milano. La richiesta si può fare anche per lettera con assegno. Il prezzo di ciascun numero arretrato è di L. 15.000 comprensivo di tutte le spese di invio.

Arrivederci al prossimo numero, fra un mese, che presenterà, fra l'altro, la nuova lezione di Linguaggio Macchina.

Editoriale VIDEO - Direttore: Antonio Lucarella - Coordinamento tecnico: Roberto Treppiedi - Hanno collaborato: Franco Longoni, Marco Affer, Alberto Barbati, Massimo Cellini, Fabrizio Iotti, Maria Russino, Aldo Campanozzi, Roberto Muraglia, Carlo Mantegazza, Stefano Milanesi, Maurizio Montevedi, Fabio Macchioni, Tito Caponio, Sebastiano Pastore, Domenico Cellamare, Roberta Di Pietro, Alessandro Vallone, Fabio Rossi, Arnaldo Restelli, Dino Ticli, Igino Zaffaina, Giovanna Zampella - Stampa: La New Graf Milano - Fotocomposizione: ERREGI Milano - reg. del Trib. di Milano n. 518 del 10/11/84 - Stampato in Italia

• NUMERI A CASO
• LA SMAZZATA

INTRODUZIONE

Eccoci giunti al 6° articolo della serie. Abbiamo ormai un bagaglio di conoscenze adeguate ad affrontare un problema complesso e a progettarne la risoluzione.
Il problema che ci proponiamo di presentare questo mese è LA DISTRIBUZIONE DI UN MAZZO DI 52 CARTE DA GIOCO A 4 giocatori. Nonostante l'apparente semplicità del problema vedremo che il programma sarà piuttosto ricco di spunti. Considerate anche che il tempo di esecuzione su un P.C. è di oltre 40 secondi: ciò significa che il problema è davvero poderoso dal punto di vista del calcolo.
Inoltre sotto la superficie del problema si nasconde un importante tema: la CASUALITÀ degli eventi coinvolti.
È proprio questa CASUALITÀ che farà da linea guida per tutto questo articolo, che come al solito, non si propone solo di presentare delle soluzioni, ma anche di suggerire delle idee di progetto ai nostri lettori.
Infine il programma BASIC che accompagna questo articolo, può essere utilizzato come punto di partenza per la più ambiziosa realizzazione di un giocatore "automatico" per qualche gioco di carte.

LA CASUALITÀ

Un programma è un insieme di istruzioni chiare e non ambigue. Lo pseudocodice insegna che la logica del programma è ben rappresentata dalle figure fondamentali e che i predicati di controllo funzionano come cerberi inflessibili.
Come possiamo allora rappresentare la CASUALITÀ propria di certi eventi naturali all'interno di un programma?
Ricordiamo che la CASUALITÀ di un evento è la proprietà di un evento di essere incerto: cioè di poter manifestarsi in un modo oppure in un altro del tutto diverso. Oppure addirittura di non manifestarsi proprio.
Un esempio tipico è la scelta di una carta in un mazzo di 52 carte. Definiamo EVENTO in questo esempio il fatto che dal mazzo venga estratta una certa carta. In una SMAZZATA (distribuzione di 52 carte a 4 giocatori) avvengono 52 di questi eventi.
Proviamo a pensare come potremmo programmare una simulazione di smazzata attraverso un programma in pseudocodice. Se perdete 5 minuti di tempo per pensarci vi renderete conto che è IMPOSSIBILE scrivere un programma che realizzi una scelta VERAMENTE a caso tra 52 carte, usando solo istru-

zioni DETERMINISTICHE, come quelle studiate fino ad ora, senza che noi programmatori siamo in grado di sapere ESATTAMENTE quale scelta viene fatta dal nostro programma. E questo distrugge la CASUALITÀ dell'evento.
Una proposta di soluzione potrebbe essere la seguente:

INIZIO. Scelta a caso tra 52 carte.
"Acquisici da tastiera il numero N"
"Scegli la N-sima carta del mazzo"
FINE.

Con l'istruzione di INPUT del numero N noi lasciamo la scelta della carta alla persona seduta davanti al PC.
Allora succede che chi si trova a immettere il numero si assume la responsabilità di scegliere a caso, e perciò si RINUNCIA a PROGRAMMARE la casualità, delegando una persona a scegliere secondo il caso.
Noi però non saremo mai soddisfatti, perché sapremo sempre che carta sceglierà per ogni N (l'N-sima è sempre la medesima).
Come uscire da questo circolo vizioso?
La soluzione esiste e consiste nel mettersi in una prospettiva diversa. In realtà la CASUALITÀ esiste solo perché noi non siamo in grado di **prevedere** con certezza se un certo evento si manifesta o no. In pratica abbiamo l'**impressione** della casualità quando non **sappiamo** prevedere quale risultato seguirà ad una azione.
Nel caso del mazzo di cui si sceglie una carta, assumere questo atteggiamento è come dire: se **sapessimo** esattamente la disposizione di tutte le carte nel mazzo dopo averle mischiate, la nostra scelta non sarebbe più CASUALE ma cosciente e voluta, così come nel caso del programma per cui scegliendo "a caso" un numero, in pratica **sapevamo** la carta selezionata.
Ecco allora la strada per risolvere l'arcano: **Si scrive un programma che generi un numero che non possiamo neanche lontanamente prevedere**, e questo numero sarà usato per scegliere una carta.
Sono sicuro che molti di voi si sentiranno presi in giro: "Ma come, - diranno, - la casualità dell'evento appartiene all'evento, e non dipende dal fatto che io sappia o no se esso capita: l'incertezza è nell'evento, non in me stesso!!"
Costoro possono essere orgogliosi di avere questa opinione filosofica, perché sono in ottima compagnia: Einstein stesso la pensava così ma solo... "in linea di principio".
Nella pratica è dimostrabile matematicamente che questa posizione e quella descritta sopra sono equivalenti nei loro effetti: sia che l'incertezza stia nell'evento, o in chi osserva l'evento, non ci sono differenze pratiche.
In uno o nell'altro dei due atteggiamenti "filosofici" le cose re-

stano nello stesso modo. Naturalmente noi scegliamo il secondo atteggiamento perché è l'unico programmabile.
La comunità dei programmatori battezza perciò CASUALE qualunque programma che generi dei numeri senza che si sappia prevedere (con certezza) quale successivo numero il programma genererà.
"Ma come" - dirà qualcuno - "neanche chi ha scritto il programma sa che numero estrarrà il programma stesso?" La risposta è NO.
Analizzando uno di questi programmi estrattori di numeri A CASO (in inglese RANDOM), si vede che però una certa legge di generazione esiste. Solo che essa è così bizzarra e complessa che nessun essere umano in modo da avere una percezione "istintiva" di una legge sottostante. Il programmatore che vede il programma estrarre un numero lo vede agire totalmene A CASO.

I NUMERI CASUALI (RANDOM) IN BASIC

In BASIC con una istruzione sola (che richiama un programma già scritto) si genera un numero con la virgola compreso tra 0 e 1.
Per esempio scrivendo:

```
10 A = RND
20 PRINT A
```

e lanciando il programma più volte si ottengono numeri tipo 0.538 oppure 0.997, insomma qualunque numero tra 0 e 1 compresi.
Per ottenere un numero in un intervallo tra, supponiamo, 0 e 50, basta moltiplicare il numero prodotto da RND per 50.
Per ottenere un numero tra 1 e 52 dovremo allora scrivere:

A = (RND* 52) + 1.

Se vogliamo un numero INTERO (cioè senza virgola) tra 1 e 52, scriveremo:

A = INT (RND*52) + 1.

Il nostro problema della scelta della carta può essere ora risolto in questo modo:

```
INIZIO. Scelta di 1 carta su 52
  "Assegna ad N un numero
  random tra 1 e 52"
  "Scegli la carta N-sima"
FINE.
```

Lanciando questo programma, il programmatore non potrà prevedere quale carta il programma estrarrà, esattamente come se incaricasse un amico di scegliere a caso un numero.

LA SMAZZATA

Il problema di distribuire 52 carte a 4 giocatori non è una semplice ripetizione di 52 scelte di una carta. C'è una notevole variante di cui tenere conto: dopo ogni estrazione bisogna trovare il modo di "eliminarla dal mazzo": essa non deve più essere estraibile.
Poi notiamo che vogliamo un sistema per rappresentare le 52 carte del mazzo, in modo da

non dover fare 52 scomode assegnazioni.
Pensiamo di rappresentare una carta mediante 2 variabili con indice: SEME(I) che rappresenta il seme della I-esima carta, e VALOR(I) che ne rappresenta il valore.
Per usare variabili numeriche (in modo da gestire i valori in modo numerico) usiamo i numeri 1,2,3 e 4 per rappresentare Picche, Cuori, Quadri e Fiori rispettivamente.
Analogamente i numeri 13,12 e 11 rappresentano Re, Donna e Asso rispettivamente.
A questo punto il caricamento (la preparazione) del mazzo ordinato si può fare così:

```
INIZIO. Caricamento del mazzo
  RIPETI
  PER I DA 1 A 4
    RIPETI
    PER J DA 1 A 13
      K = (I-1)*13 + J
      SEME (K) = I
      VALOR (K) = J
    FINE
  FINE
FINE.
```

Questo programma inserisce in SEME(K) e VALOR(K) tutte le carte ordinate da Picche a Fiori. La gestione di K è divertente e lascio al lettore il compito di capirne la logica (suggerimento: in questo modo K assume i valori da 1 a 52 in sequenza).
Dopo aver creato il mazzo pensiamo alla distribuzione:

```
INIZIO. Distribuzione
  RIPETI
  PER I DA 1 A 13
    RIPETI
    PER J DA 1 A 4
      "Scegli a caso una carta
      tra quelle rimaste"
      "Dai la carta al giocatore
      J"
      "Elimina la carta estratta
      dal mazzo.
    FINE_RIPETI
  FINE_RIPETI
FINE.
```

(Osservate la scelta del giocare J-simo, in quanto J oscilla tra 1 e 4).
Per ogni giocatore dobbiamo pensare a come memorizzare che ha ricevuto la carta estratta.

Pensiamo a una variabile a doppio indice: il primo indica il giocatore, il secondo il numero progressivo della sequenza di carte a lui distribuita.
Però per ogni carta noi vogliamo sapere sia il valore che il seme. Perciò avremo una coppia di variabili a 2 indici: chiamiamole GSEM (H,L) e GVAL (H,L). Per esempio se GSEM (4,10) = 2 e GVAL (4,10) = 1 allora il giocatore 4 ha come 10-ma carta in mano un asso di Quadri. Chiaro? Un altro esempio: GSEM (3,2) = 1 e GVAL (3,2) = 11 significa che la 2ª carta del 3° giocatore è un Fante di Picche.
Ecco come si riscrive il nostro programma, con queste ipotesi:

```
INIZIO. Distribuzione.
  RIPETI
  PER I DA 1 A 13
    RIPETI
    PER J DA 1 A 4
      "Scegli una carta a caso:
```

```
      la carta di indice X"
      GSEM (J,I) = SEME(X)
      GVAL (J,I) = VALOR(X)
      "Elimina la carta dal
      mazzo"
    FINE_RIPETI
  FINE_RIPETI
FINE.
```

Osservate il gioco di J ed I, che contano i giocatori ed i giri rispettivamente.
Siamo arrivati al problema cruciale: come scegliere nel mazzo una carta tra 52 al primo giro una tra 51 al secondo ... fino ad 1 tra 2 al penultimo giro?
Bisogna introdurre una variabile P al posto del 52 nell'istruzione descritta al paragrafo precedente:

```
X = INT(RND*(P)) + 1
```

e poi far decrescere P da 52 a 1 levando 1 ad ogni giro.
L'ultimo problema rimasto è: come eliminare dal mazzo una carta già scelta? Ecco un'idea. Una volta scelta la carta X ed assegnati i valori al giocatore J, giro I, facciamo un riordinamento dei vettori SEME(N) e VALOR(N). Spostiamo tutto il mazzo in alto di una carta di un posto a partire dalla carta X + 1. In questo modo, alla prossima scelta tra 1 e (P-GIRO) cadremo sempre su una carta non ancora scelta.
Meglio ancora se mettiamo Ø sia in seme che in valore nell'ultima carta del mazzo. Così pian piano i vettori si riempiono di zeri dal fondo, indicando le carte eliminate.
Ecco lo pseudocodice:

```
INIZIO_ Elimina carta X dal
mazzo.
  RIPETI
  PER N DA 1 A 52
    VALOR(N) = VALOR(N + 1)
    SEME(N) = SEME(N + 1)
  FINE_RIPETI
FINE.
```

(Per usare con eleganza questo trucco, conviene mettere una 53 carta fittizia nel mazzo il cui valore e seme valgono Ø).
Con questo i pezzi del nostro programma sono tutti a posto. Basta creare il programma di stampa delle carte dei giocatori al termine della smazzata.
Per fare questo usiamo un vettore di decodifica: per ogni numero di VALOR assegnamo un carattere ASCII che ne rappresenta la carta. Lo stesso per il SEME.
Chiamiamo il primo vettore RV$, ed il secondo RS$.
Allora possiamo anche pensare di raggruppare le carte per seme, in output. Otteniamo:

```
INIZIO Stampa.
  RIPETI
  PER I DA 1 A 4
    Stampa "Giocatore" I
      Stampa seme: RS$(J)
    PER K DA 1 A 13
      SE GSEM (I,K) = J
      ALLORA Stampa RV$
      (GVAL(I,K))
      FINE_SE
    FINE RIPETI
  FINE_RIPETI
FINE.
```

Ecco fatto, il programma è progettato. Chi vuole può divertirsi a realizzarlo per conto proprio.

Per chi si è già...stufato il programma BASIC è già pronto in figura 1.
Memorizzatelo e fatelo girare. Il risultato è simile a quello mostrato in figura 2, ma leggete attentamente la didascalia.
L'istruzione 131 apparentemente non ha molto senso. Ma provate a levarla...avrete l'emozione di attendere con "suspense" che il programma termini 40 secondi (un'eternità) di elaborazione.
Arrivederci

M.S.

```
10 DIM SEME(53),VALOR(53),GVAL(4,13),GSEM(4,13),RS$(4),RV$(13)
15 RS$(1)="P":RS$(2)="C":RS$(3)="Q":RS$(4)="F"
16 RV$(1)="1":RV$(2)="2":RV$(3)="3":RV$(4)="4"
17 RV$(5)="5":RV$(6)="6":RV$(7)="7":RV$(8)="8"
18 RV$(9)="9":RV$(10)="10":RV$(11)="J":RV$(12)="Q":RV$(13)="K"
20 REM Smazzata
30 GOSUB 2000  'Richiama Carica-mazzo
40 P=52        'numero di carte rimanenti nel mazzo
50 FOR I=1 TO 13
60     FOR J=1 TO 4
80         X=INT(RND*(P))+1
90         GSEM(J,I)=SEME(X)
100        GVAL(J,I)=VALOR(X)
110        GOSUB 3000     ' richiama elimina carta
120        P=P-1
130    NEXT J
131   PRINT "GIRO NUMERO ..." I
140 NEXT I
150 ' Stampa le carte smazzate
160 FOR I=1 TO 4
170   PRINT " GIOCATORE Numero " I
180   FOR J=1  TO 4
190     PRINT RS$(J); "  :";
200     FOR K=1 TO 13
210      IF GSEM(I,K)=J THEN PRINT RV$(GVAL(I,K));" ";
220     NEXT K
225     PRINT
230   NEXT J
240 NEXT I
250  STOP
2000  ' carica il mazzo di carte
2010  FOR I=1 TO 4
2020    FOR J=1 TO 13
2030      K=(I-1)*13+J
2040      SEME(K)=I
2045     VALOR(K)=J
2060    NEXT J
2070 NEXT I
2075  SEME(53)=0 : VALOR(53)=0
2080  RETURN
3000 ' elimina una carta dal mazzo
3010  FOR N=X TO 52
3030     VALOR(N)=VALOR(N+1)
3035     SEME(N)=SEME(N+1)
3040  NEXT N
3070  RETURN
```

```
 GIOCATORE Numero  1
P  :9 1
C  :4 5 8
Q  :7 K 1 9
F  :4 8 10 K
 GIOCATORE Numero  2
P  :10 7 6 J K
C  :1 10 7 K
Q  :J
F  :5 3 1
 GIOCATORE Numero  3
P  :2 5 8 Q
C  :3 J
Q  :8 10 6 4 Q
F  :Q 2
 GIOCATORE Numero  4
P  :4 3
C  :Q 2 6 9
Q  :3 2 5
F  :7 9 J 6
Break in 250
Ok
```

Ecco l'esempio di una smazzata del programma SMAZZ. Chi scriverà il proprio programma in BASIC copiando il testo qui presentato in fig. 1, ben difficilmente riuscirà ad ottenere esattamente questa smazzata: c'è la stessa probabilità che si ottengano 2 distribuzioni successive o identiche di carte!

IL LINGUAGGIO MACCHINA

Per imparare a conoscere il linguaggio macchina è utile puntualizzare i concetti di elettronica digitale, TTL, sistema binario e sistema esadecimale. È opportuno, inoltre, approfondire la conoscenza del microprocessore. A questa materia dedichiamo la prima lezione.

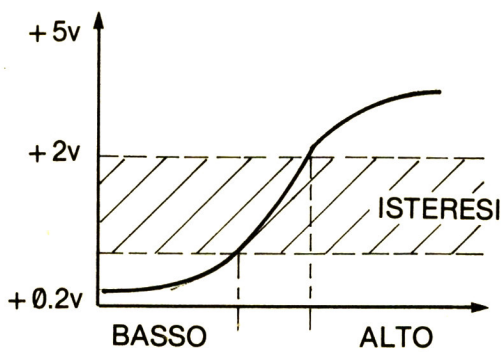
1.1 L'ELETTRONICA DIGITALE

Prima di avventurarci nel cuore vero e proprio del microprocessore (il cervello del computer), penso sia utile, per chi affronta per la prima volta il l/m, dare qualche fondamentale nozione di elettronica digitale, che servirà in seguito a rendere più semplice e completo l'apprendimento del linguaggio macchina vero e proprio. Esistono principalmente due campi dell'elettronica, quello digitale e quello analogico. Mentre nell'elettronica analogica si considerano le variazioni di differenze di potenziale (i famosi volts), in elettronica digitale si hanno due sole ben definite differenze di potenziale; vale a dire, in elettronica analogica un dispositivo può generare un campo continuo di volts (per es. da 0 a 20 volts), mentre un dispositivo digitale può generare solo due ben definite differenze di potenziale, che variano a seconda della famiglia cui il dispositivo appartiene, ma che all'interno di tale famiglia sono ben definite e standard. Per fare un esempio possiamo dire che un integrato digitale (quei strani contenitori plastici neri con due file di piedini sporgenti che si trovano all'interno di ogni computer e che contengono uno o più dispositivi digitali) della famiglia dei TTL, può generare solo due valori di tensione in uscita, 0 volts oppure + 5 volts. Naturalmente esistono più famiglie di integrati digitali quali DTL, ECL, TTL, RTL ecc. L'appartenenza di un integrato ad una data famiglia dipende esclusivamente dall'architettura interna e dagli standard di tensione necessari al dispositivo stesso, ma evitiamo questo discorso, che sarebbe troppo vasto e complesso da spiegare in poco spazio.

1.2 LA FAMIGLIA DEI TTL

La famiglia di integrati che possono principalmente interessarci è quella dei TTL (Transistor-Transistor Logic), in quanto questa è la più usata nella progettazione ed interfacciamento di computers. Abbiamo già accennato al fatto che un dispositivo TTL può generare un segnale di 0 volts o + 5 volts, ma questo avviene solo in teoria, in quanto a causa di vari fattori interni ed esterni, tali

tensioni possono variare di una piccola percentuale. Quello che comunque ci interessa maggiormente è il fatto che il dispositivo è in grado di generare due tensioni ben separate tra loro da un intervallo che viene chiamato 'isteresi' del dispositivo.
Nel grafico si nota come l'impulso di tensione che in teoria dovrebbe essere di +5 volts (tale livello lo chiamiamo 'alto') possa variare da un minimo di +2 volts ad un massimo di +5 volts, mentre l'impulso di tensione che dovrebbe essere di 0 volts (questo livello lo chiamiamo 'basso') può arrivare ad un massimo di +0.8 volts.
Se noi al livello basso di tensione attribuiamo il numero 0 ed al livello alto il numero 1, otteniamo che un dispositivo digitale è in grado di generare un numero che ha valore 0 o 1, che viene chiamato 'BIT'. Questa è appunto la spiegazione per cui tutti i computers si basano sul sistema di numerazione binario, in quanto questo permette operazioni basandosi su numeri formati da due soli simboli, 0 oppure 1.



1.3 IL SISTEMA BINARIO

Ogni sistema di numerazione è caratterizzato da un numero b, detto base del sistema, che equivale al numero di simboli disponibili nel sistema dato per rappresentare un numero. Nel sistema decimale, b equivale a 10, in quanto abbiamo a disposizione dieci simboli (da 0 a 9), le cui combinazioni ci permettono di rappresentare qualsiasi numero. Una regola da tener ben presente è che se si hanno n cifre formanti un numero, le combinazioni possibili tra queste n cifre sono uguali alla base b elevata ad n (b^n), infatti in base decimale con quattro cifre noi possiamo rappresentare $10^4 = 10000$ combinazioni diverse di numeri (da 0000 a 9999). Inoltre una di queste combinazioni, per esempio 3658, è scomponibile nel seguente modo:

$3658 = 3 \times 1000 + 6 \times 100 + 5 \times 10 + 8$

ossia:

$3658 = 3 \times 10^3 + 6 \times 10^2 + 5 \times 10^1 + 8 \times 10^0$

Riconsiderando la b e la n della discussione precedente notiamo che $10^2 = 10^{n-1}$, $10^2 = 10^{n-2}$, e così via; per cui:

$3658 = 3 \times 10^{n-1} + 6 \times 10^{n-2} + 5 \times 10^{n-3} + 8 \times 10^{n-4}$

Consideriamo ora le cifre 3,6,8,5, come coefficienti, e chiamiamoli a. Ne segue che un qualsiasi numero A è rappresentabile con la seguente formula:

$A = a_{n-1} \times b^{n-1} + a_{n-2} \times b^{n-2} + ...a_1 \times b^1 \times a_0 \times b^0$

Quest'ultima formula (polinomiale) è applicabile a qualsiasi sistema di numerazione, per cui un numero binario di 4 cifre (1011) è scomponibile come:

$1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

Tale calcolo risolto dà come risultato 11 in base decimale (11_{10}). Possiamo così convertire facilmente una cifra da binario a decimale, e di conseguenza un dispositivo digitale può rappresentare facilmente anche un numero decimale. Se per esempio abbiamo tre diverse linee elettriche, ed in ognuna di queste abbiamo o 0 volts o +5 volts abbiamo una combinazione binaria. Infatti se 0 volts = 0 e +5 volts = 1 segue che:

+5V _____ 1
+0V _____ 0
+0V _____ 1

Da ciò segue che tre linee elettriche possono rappresentare, come da figura, il numero binario 101_2 (101_2 per dire 101 in base 2), equivalente al numero decimale 5_{10} (5_{10} per dire 5 in base 10). Anche in questo caso, come nel sistema decimale, date n cifre si possono calcolare il numero di combinazioni possibili elevando la base b ad n,b. Per cui in un sistema binario con tre cifre si possono avere un massimo di $2^3 = 8$ combinazioni, con quattro cifre un massimo di $2^4 = 16$ combinazioni.

1.4 IL SISTEMA ESADECIMALE

Imparato a portare un numero a binario a decimale, sorge ora un problemino; infatti un programmatore che per la prima volta si avventura nel linguaggio macchina, nota che un numero decima-

le è rappresentabile tramite quattro bits, ma le combinazioni di tali unità binarie sono $2^4 = 16$, quindi ben al di sopra delle dieci che richiede il sistema decimale, infatti rappresentando tutte le combinazioni di quattro bits e mettendovi a fianco la corrispondente conversione decimale otteniamo:

binario	decimale
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	?
1011	?
1100	?
1101	?
1110	?
1111	?

(figura 1.4/1)

Si vede quindi che non esiste alcun simbolo per rappresentare le combinazioni superiori a nove (qualcuno certo penserà a 10,11,12 ..., ma ricordo che questi sono numeri e non simboli).
Da ciò si rende necessaria l'introduzione di altri simboli, che insieme ai dieci simboli decimali costituiranno il sistema esadecimale (a base 16); tali simboli sono le prime sei lettere dell'alfabeto (A-F), per cui si ottiene la seguente tabella:

binario (base 2)	decimale (base 10)	esadecimale (base 16)
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	'10'	A
1011	'11'	B
1100	'12'	C

1101	'13'	D
1110	'14'	E
1111	'15'	F

(figura 1.4/2)

Se qualcuno ancora pensasse che i numeri 10,11,... ecc. avessero risolto tale problema altrettanto bene quanto l'esadecimale, faccio notare che se abbiamo un numero binario a otto bits, noi lo possiamo suddividere in due numeri in due numeri da 4 bits ciascuno (nibble).
per esempio:
10110010 si può dividere in 1011 e 0010.

Rappresentando questi due nibbles con due numeri decimali risulterebbe:

$1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11_{10}$
$0010 = 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 2_{10}$

Accoppiando in seguito i due numeri ottenuti si ha come risultato 112.
Se calcoliamo ora il vero valore del numero ad 8 bits otteniamo.

$10110010 = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^0 = 178_{10}$

Notiamo quindi che in decimale il numero ricavato dai nibbles (112) non corrisponde al reale valore del byte. Se usiamo il sistema esadecimale sui nibbles otteniamo il numero B2H(H stà per esadecimale).
Ricavandoci ora con la formula polinomiale il corrispondente valore decimale di B2 otteniamo:

$B2_{16} = B_{10} \times 16^1 + 2_{16} \times 16^0$

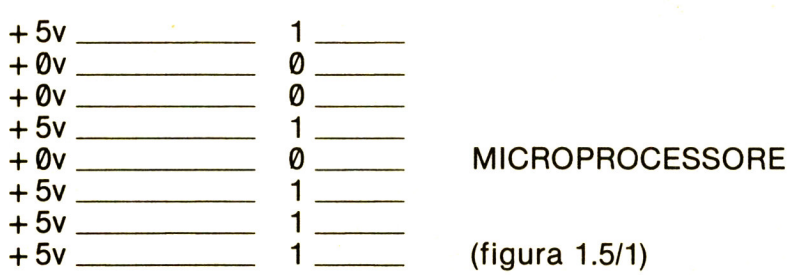
Ricordando che B_{16} corrisponde al valore 11_{10} (fig. 1.4/2) segue:

$B2_{16} = 11 \times 16^1 + 2 \times 16^0 = 178_{10}$

che corrisponde al valore effettivo del byte.
Quindi col sistema esadecimale noi possiamo rappresentare direttamente numeri binari formati da multipli di 4 bits tramite simboli, mentre col sistema decimale noi dovremmo ricavarci il valore del numero tramite formula aritmetiche.

1.5 I MICROPROCESSORI

Il menzionare bytes, bits e TTL non è stato fino ad adesso senza scopo, infatti tutto ciò serve ad introdurvi al uP (Microprocessore) ad un livello superiore a quello raggiunto da coloro che hanno esperienza nel solo campo informatico. Il mio scopo è quello di farvi capire oltre il linguaggio macchina le sue interconnessioni con la parte 'hard' (la parte tecnica vera e propria).
Un uP ha il compito di svolgere tutte le funzioni matematiche e di trasmissione dati all'interno di un computer. Esternamente il uP si presenta come un normale integrato TTL di notevoli dimensioni, il cui contenitori (DIP), oltre che plastico, è spesso anche ceramico. All'interno di tale contenitore esiste il 'chip', una piastrina di silicio di piccolissime dimensioni (qualche mm quadrato). Tale chip è connesso elettricamente ai piedini (pins) che escono ai due lati del DIP. Tali piedini hanno la funzione di collegare elettricamente l'integrato stesso al circuito stampato (la piastra con piste in rame su cui sono montati i componenti) in modo da creare varie connessioni tra tutti gli integrati. L'insieme di tali connessioni è detto 'rete logica' e permette la trasmissione dei dati e la sincronizzazione tra i vari integrati o dispositivi esterni (periferiche).
Ogni pin di un uP svolge varie funzioni. Una parte dei piedini di un uP è dedicata a ricevere ed a trasmettere dati o istruzioni da e per la CPU (un altro modo di definire il microprocessore, da Central Processing Unit). Tale insieme di piedini è chiamato 'Data Bus' (via per i dati) ed il funzionamento è abbastanza semplice. Poniamo l'esempio che tale 'Bus' sia formato da 8 pins (in tal caso il uP si dice ad 8 bits) e forniamo a tali piedini una combinazione di livelli alti e bassi (0 volts e +5 volts). Ogni combinazione che noi forniamo sul bus può corrispondere ad un dato numerico o istruzione che la CPU deve elaborare. Spiegando ciò tramite uno schema otteniamo la figura 1.5/1, in cui troviamo 8 linee elettriche (il Data Bus) che vanno a connettersi al uP.



(figura 1.5/1)

La combinazione dei segnali dà come risultato un numero binario, (10010111 in fig. 1.5/1) che per il uP ha un significato ben preciso e che può essere un dato numerico su cui lavorare o un'i-

struzione da eseguire. Da ciò si può ricavare che:

1) Il uP (di conseguenza il linguaggio macchina) non fà distinzioni tra dati numerici su cui lavorare o istruzioni da eseguire, infatti entrambi sono combinazioni binarie.

2) Il linguaggio macchina si riduce sempre ad un insieme di numeri, spetta al programmatore ordinarli in modo che il uP esegua il programma in modo corretto (vedremo poi come).

Per rendere più evidente di come i numeri possano rappresentare delle istruzioni prendiamo ad esempio il numero 76_{16} (118_{10}). Tale numero, inviato sul bus dati di una CPU Z-80 (Z-80 è il nome), ferma ogni elaborazione di quest'ultima, la quale rimane in attesa che avvenga una certa condizione, oppure il dato $F8_{16}$ (243_{10}), sempre su Z-80, impedisce che la CPU venga interrotta da un dispositivo esterno durante una elaborazione dati.
Naturalmente la corrispondenza tra codici numerici ed istruzione non è universale per tutti i uP, ma varia a seconda del tipo della CPU. Inoltre i dati inviabili su una data bus di un uP varia a seconda delle dimensioni del bus stesso, da qui la denominazione di un uP a 8,16,32 bits, a seconda, appunto, l'accettare le combinazioni di 8,16,32 linee elettriche in entrata come dati.
Ecco i nomi di alcuni tra i più usati microprocessori:

6510	8 bits
Z-80	8 bits
8080	8 bits
8085	8 bits
68000	16 bits
TMS 9900	16 bits
Z 8000	16 bits

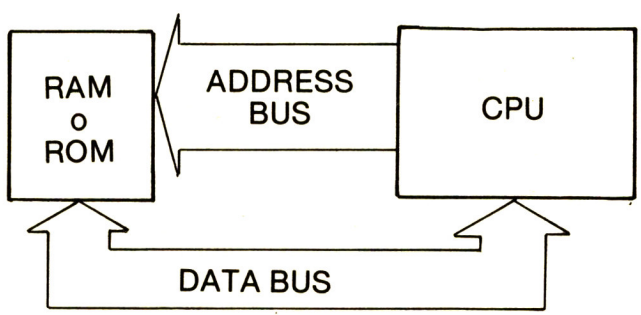
(figura 1.5/2)

A questo punto, capito come si invia una istruzione ad un uP bisogna spiegare come quest'ultimo può trovare la sequenza di istruzioni da eseguire. Oltre al 'Data Bus', esiste un 'Address Bus' la cui costituzione è identical al 'Data Bus', ma dove la direzione dei dati è solo verso le memorie. Le combinazioni di Bits su questo bus, invece di rappresentare delle istruzioni, vanno a pilotare delle memorie esterne. Queste sono formate da un insieme di piccole celle che hanno la capacità di memorizzare un byte e che sono numerate. Le varie combinazioni sull'Address bus servono a selezionare una di queste celle. Una volta selezionata una cella, il contenuto di questa è posto dalla memoria sul Bus dati, permettendo alla CPU di leggerlo. Naturalmente la CPU, una volta selezionata una cella, può anche scrivervi all'interno, ed è compito della memoria raccogliere il dato da memorizzare dal Data bus. Notiamo quindi una differenza tra Data ed Address

bus. Infatti nel primo i dati possono venire trasmessi dalla memoria alla CPU e vice versa. Per tal motivo tale bus è detto 'Bidirezionale' (con due direzioni di flusso dati), al contrario dell'Address bus, che si dice 'unidirezionale', in quanto il flusso di indirizzi va solo verso la memoria. Ricordiamo che l'Address bus può variare in grandezza (come il Data bus) a seconda del uP cui appartiene. Usualmente un uP ad 8 Bits ha un Address bus di 16 Bits, la cui combinazione può indirizzare $2^{16} = 65536$ celle di memoria diverse. Tale numero è anche noto come 64K, facendo corrispondere 1K a $2^{10} = 1024$ celle di memoria. Dunque più è ampio un Address bus, maggiore è la capacità di memoria accessibile da una CPU.
Ecco a questo punto le dimensioni degli Address Buses dei precedenti microprocessori:

uP	Data Bus	Addr.Bus
6510	8 bits	16 bits
Z-80	8 bits	16 bits
8080	8 bits	16 bits
8085	8 bits	16 bits
68000	16 bits	24 bits
TMS 9900	16 bits	16 bits
Z-8000	16 bits	16/23 bits

Inoltre eccovi uno schema flusso dati delle connessioni tra memoria esterna e CPU:



La freccia dell'Address bus indica un unico flusso dati (gli indirizzi) verso la memoria, mentre quella doppia della Data bus indica le due possibili direzioni del flusso di dati.
Detto questo concludo questa prima parte sul 1/m, sperando di essere stato abbastanza chiaro con questa introduzione elettronica, rendendovi forse per adesso l'apprendimento un po' complesso, ma sicuro di esservi stato utile per futuri capitoli del corso sul l/m e su tutti i uP in generale. 1 - continua

A.V.S.

LE FUNZIONI PRINCIPALI

La volta scorsa abbiamo esaminato le istruzioni fondamentali del BASIC e abbiamo anche parlato di variabili indirizzate, stavolta ci occuperemo invece di alcune istruzioni di uso meno frequente e di tutte le funzioni di cui dispone il BASIC.
Come certo ricorderete, per porre il computer in condizione di ricevere dati durante l'esecuzione di un programma abbiamo sempre usato l'istruzione INPUT, ma essa non è l'unica che permette di interagire con l'operatore; esiste infatti anche l'istruzione GET, la quale però, diversamente da INPUT, non riceve un insieme di dati seguiti dalla pressione di RETURN, ma "legge" quale tasto è stato premuto e ne assegna il valore alla variabile specificata. Per esempio, l'istruzione:

```
GET H$
```

riconosce quale tasto è stato premuto e ne assegna il carattere corrispondente alla variabile H$. Se nessun tasto era premuto al momento dell'esecuzione di questa istruzione alla variabile a$ sarà assegnato un valore nullo. Come avrete capito, questa istruzione è molto comoda in diverse circostanze; se per esempio volessimo fermare temporaneamente il programma (magari in attesa che l'utente legga qualcosa) potremmo procedere come segue:

```
100 GET H$
105 IF A$ = "" THEN GOTO
    100
```

In questo modo la linea 100 assegna a H$ il carattere che è stato letto dalla tastiera e se risulta che nessun tasto è stato premuto (stringa nulla) la linea 105 rimanda alla 100 finché verrà premuto un qualsiasi tasto.
Con l'istruzione GET si possono specificare anche variabili numeriche, ma fate attenzione perché se il tasto premuto non sarà un numero il computer si arresterà mostrandovi un messaggio d'errore.
Per restare in tema di variabili, argomento peraltro assai ampio, citiamo un altro modo di assegnazione di valori a una variabile, stavolta però interno al programma e dove i dati interessati devono essere scritti durante la stesura del programma stesso. L'istruzione a cui ci riferiamo è READ che in inglese significa proprio LEGGI.
Se noi in un programma dobbiamo usare grandi quantitativi di dati che rimangono sempre tali e non devono quindi essere frequentemente aggiornati possiamo porre tali dati in un programma grazie all'istruzione DATA seguita appunto dai dati che ci interessano i quali possono essere numerici o alfanumerici e separati fra loro da una virgola. Questi dati saranno poi letti in sequenza dall'istruzione

READ seguita da un nome di variabile a cui si desidera assegnare il dato letto; fate attenzione che il tipo di variabile che usate corrisponda effettivamente al tipo di dato che deve leggere, inoltre ricordate che le stringhe devono essere chiuse fra apici. Vediamo ora un semplice esempio, cercate di capire da soli cosa fa e in che modo.

```
10 FOR A=1 TO 7
20 READ S$
30 PRINT S$
40 NEXT A
50 DATA "LUNEDI",
   "MARTEDI"
60 DATA "MERCOLEDI",
   "GIOVEDI"
70 DATA "VENERDI",
   "SABATO"
80 DATA "DOMENICA"
```

Come probabilmente avrete capito, la linea 10 inizia un ciclo che va da 1 a 7 all'interno del quale si trova una istruzione che legge (READ) i valori contenuti nelle linee di DATA e li assegna alla variabile stringa S$ la quale viene stampato subito dopo. Come potete vedere, nelle linee di DATA che seguono il programma principale abbiamo messo il nome dei giorni della settimana che sono un tipico esempio di dato costante; per chiarezza abbiamo inserito solo due giorni per ogni istruzione DATA, ma è possibile porne molti di più, a condizione che la lunghezza del tutto non superi gli 80 caratteri che rappresentano la massima lunghezza possibile di ogni linea di programma. Non ha comunque nessuna importanza se i dati si trovano su linee diverse, in quanto l'istruzione READ punta sempre al dato successivo a quella appena letto, comunque essa si trovi.
Se noi però dovessimo leggere più volte gli stessi dati non potremmo farlo se non ricorrendo a una speciale istruzione chiamata RESTORE, la quale resetta i puntatori al primo dato della prima istruzione DATA, permettendo quindi di rileggere il tutto quante volte vogliamo. Vediamo un semplice esempio:

```
10 PRINT "NUMERI PRIMI"
20 FOR A=1 TO 5
30 READ N
40 PRINT N
50 NEXT A
60 GET P$
65 IF P$="" THEN GOTO 60
70 RESTORE:PRINT CHR$(147)
75 GOTO 10
90 DATA 1,3,5,7,11
```

Questo programmino stampa i primi cinque numeri primi che sono memorizzati in linea 90, con un ciclo simile al precedente, dopodiché le linee 60 e 65 attendono la pressione di un tasto al che vengono eseguite le linee 70 e 75 che resettano i puntatori di dato e cancellano lo schermo, tornando quindi alla linea 10.
Concludiamo osservando che le linee di DATA non si devono trovare necessariamente alla fine del programma, ma poiché non eseguono nessuna operazione sarebbe inutile porle nel mezzo del programma e causerebbero solo un rallenamento

nei tempi di esecuzione, inoltre si preferisce sistemarle alla fine anche per ovvie ragioni estetiche e per non pregiudicare la facile lettura del programma da parte di un'altra persona.
Esamineremo ora un tipo molto particolare di istruzioni che potremmo definire di "salto", ma data la loro particolare struttura abbiamo preferito aggirarle nella prima puntata. Stiamo parlando delle due istruzioni ON GOTO e ON GOSUB.
Queste due istruzioni causano un salto a uno dei tanti possibili punti specificati, la scelta del quale dipende dal valore della variabile che segue la particella ON. Se per esempio scriviamo:

```
ON A GOTO 100,500,3000
```

Se la variabile A contenesse 1 il programma salterebbe alla linea 100, se A contenesse 2 salterebbe alla linea 500, mentre sa A contenesse 3 salterebbe alla linea 3000. Come abbiamo già detto, la scelta del numero di linea a cui saltare è subordinato al valore della variabile usata come controllo. In pratica questa struttura sostituisce vantaggiosamente una serie di IF...THEN. Se la variabile di controllo contenesse 0 o un numero superiore ai numeri di linea specificati il programma proseguirebbe all'istruzione successiva; se la variabile fosse invece negativa verrebbe visualizzato un messaggio d'errore, mentre se il contenuto della variabile non fosse intera, verrebbe considerata la sola parte intera, quindi 3.9 sarebbe considerato come 3.
L'applicazione pratica cui queste istruzioni tendono in modo prevalente è il controllo delle selezioni fatte in un menù; vediamone un esempio tipico.

```
10 PRINT CHR$(147)
15 PRINT "1-INSERIMENTO"
20 PRINT "2-STAMPA"
25 PRINT "3-FINE"
30 GET A
35 ON A GOTO 50,60,70
40 GOTO 30
50 INPUT"NOME?";N$
55 GOTO 10
60 PRINT N$
65 GOTO 10
70 END
```

In questo semplice programmino dimostrativo le linee da 10 a 25 puliscono lo schermo e stampano un menù costituito da tre opzioni; successivamente l'istruzione GET A attende la pressione di un tasto numerico e, a seconda che venga premuto 1, 2 o 3 il programma salta a una diversa linea. Se noi volessimo inserire un nome rememremmo il tasto 1 che causerebbe un salto alla linea 50 dove troviamo appunto una richiesta di input; la linea successiva salta all'inizio del programma facendo in modo che lo schermo sia pulito e venga ripresentato il menù. Notate che se noi premessimo un tasto numerico diverso da 1, 2 o 3 l'istruzione ON GOTO non verrebbe eseguita e la linea successiva causerebbe un ritorno alla linea 30.
Quanto esposto finora è valido anche nel caso dell'istruzione ON GOSUB, solo che quest'ulti-

ma salta a una subroutine specificata, la quale deve terminare con un RETURN che restituisce il controllo alla linea successiva a quella di chiamata. Per meglio chiarire il concetto, vediamo come diventerebbe il programma precedente se usassimo l'istruzione ON GOSUB invece di ON GOTO.

```
10 PRINT CHR$(147)
15 PRINT "1-INSERIMENTO"
20 PRINT "2-STAMPA"
25 PRINT "3-FINE"
30 GET A
35 ON A GOSUB 50,60,70
40 GOTO 10
50 INPUT "NOME?";N$
55 RETURN
60 PRINT N$
65 RETURN
70 END
```

Come potete notare ogni routine (tranne l'ultima) termina con una istruzione RETURN che trasferisce il controllo alla linea 40, la quale a sua volta salta alla linea 10 permettendo di visualizzare nuovamente il menù. Notiamo che se in questo caso non premessimo alcun tasto, o premessimo numeri non validi, la linea 40 salterebbe di nuovo all'inizio del programma stampando continuamente il menù; in questo caso quindi sarebbe meglio porre un controllo fra le linee 30 e 35 per verificare se qualche tasto è stato premuto. Vedremo in seguito, parlando delle funzioni, come fare per porre un controllo che accetti solo la pressione di tasti numerici.
Bene, a questo punto abbiamo praticamente terminato l'analisi delle istruzioni BASIC disponibili sul COMMODORE 64 e prima di passare a esaminare le molteplici funzioni di cui esso dispone non ci resta che imparare a "saluare" e "caricare" i programmi su una normale musicassetta.
Se avete in memoria un programma, non devete fare altro che introdurre nel registratore collegato al vostro computer una cassetta vergine (o comunque che non usate) e scrivere poi il comando SAVE "nome", dove "nome" può essere qualunque stringa composta al massimo di 17 caratteri che rappresentano il nome del programma. Battete ora il tasto RETURN e vedrete apparire la scritta:

PRESS RECORD AND PLAY ON TAPE

Che significa:

PREMI I TESTI RECORD E PLAY DEL REGISTRATORE

Eseguite e vedrete lo schermo divenire azzurro, segno che il computer sta "salvando" il programma su nastro. Terminata questa operazione apparirà il messaggio READY. A questo punto sarebbe sempre meglio verificare la corretta registrazione dei dati per evitare spiacevoli sorprese; ciò si può fare con il comando VERIF" "nome", il quale confronta quanto presente su nastro con i dati presenti in memoria e, se trova degli errori ve lo segnalerà (in questo caso ripetere l'operazione

di salvataggio), in caso contrario apparirà il messaggio ON. Ora il vostro (o altrui) programma è memorizzato permanentemente sulla cassetta e potrete ricaricarlo quando vorrete con il comando LOAD "nome" che carica in memoria il programma chiamato "nome".
All'esecuzione di questo comando apparirà la scritta

PRESS PLAY ON TAPE

Eseguite premendo il tasto PLAY del registratore e attendete alcuni secondi trascorsi i quali apparirà la scritta

FOUND: nome

Che sta a significare che il computer ha trovato sul nastro un programma chiamato "nome"; se esso coincide con il nome da voi specificato nel comando LOAD il programma sarà caricato, altrimento il computer proseguirà la sua ricerca. Se dimenticate il nome del programma che desiderate caricare scrivete semplicemente LOAD, senza specificare alcun nome; in questo modo sarà caricato il primo programma trovato. Terminato il caricamento (il cui tempo varia proporzionalmente alla lunghezza del programma), scrivete LIST seguito come sempre dalla pressione del tasto RETURN e vedrete il LISTATO (così si chiama l'insieme delle istruzioni e numeri di riga) scorrere velocemente sullo schermo, specificando un numero dopo il comando LIST sarà visualizzata solo la linea specificata, mentre scrivendo LIST seguito da un numero di riga e un trattino saranno visualizzate tutte le linee a partire da quella specificata; se scriviamo invece prima il trattino e poi il numero di linea saranno mostrate tutte le linee fino a quella specificata. Tutte queste opzioni sono molto utili per correggere o modificare i programmi.
Passiamo ora a descrivere dettagliatamente tutte le funzioni che il BASIC del 64 mette a disposizione.
La prima funzione che incontriamo è ABS (), la quale restituisce il valore assoluto del numero o variabile numerica posta fra parentesi; in altre parole priva il numero dell'eventuale segno che lo precede, quindi l'istruzione

PRINT ABS (-91.4)

produrrebbe la stampa di 91.4. Questa particolare funzione può rivelarsi utile nel caso in cui si desideri sapere se una variabile contiene un numero positivo o negativo; basta infatti procedere nel modo seguente:

IF A = ABS(A) THEN ...

In questo modo le istruzioni che seguono THEN saranno eseguite solo se A contiene un valore positivo. Cercate di capirne da soli la ragione.
La funzione SGN () determina invece direttamente il segno di una variabile posta fra parentesi e fornisce come risultati:

1 se positivo
-1 se negativo
Ø se nullo

Sapendo questo, possiamo riscrivere la condizione dell'esempio precedente nella seguente forma:

```
IF SGN(A) = 1 THEN ...
```

Infatti, il risultato della funzione sarà 1 solo se la variabile contiene un numero positivo.
Un'altra funzione matematica molto usata è INT(). Questa funzione fornisce la parte intera del numero posto fra parentesi. L'arrotondamento è sempre effettuato all'intero più basso, quindi l'istruzione

```
PRINT INT (10.9)
```

darebbe come risultato 10 mentre l'istruzione

```
PRINT INT (-20.1)
```

restituirebbe il valore -21. Questa particolare funzione è spesso usata in abbinamento con RND(), la quale permette di generare numeri casuali o pseudocasuali. Il numero posto fra parentesi non ha significato, ma influisce sulla generazione dei numeri casuali a seconda che esso sia positivo, negativo o nullo. Nel primo caso si ottengono numeri casuali partendo da un "seme" generato all'accensione del computer; nel secondo caso si ottiene invece sempre la stessa sequenza di numeri, mentre nel terzo caso i numeri vengono ricavati direttamente dal CLOCK del 64 (il suo orologio interno). Normalmente si usa la prima funzione che genera comunque solo numeri compresi fra 0 e 1 (mai 1), ma con qualche piccolo artificio di programmazione si può fare in modo di produrre sequenze di numeri compresi entro i limiti che più ci interessano, infatti se noi moltiplichiamo il risultato della funzione RND(1) per il massimo numero che ci interessa, otterremo proprio numeri compresi fra 0 e il numero specificato. Vediamo un esempio:

```
A = RND(1)*20
```

In questo modo otterremo che la variabile A contenga solo numeri compresi fra 0 e 19. Ma se noi desiderassimo che la sequenza di numeri non inizi da 0 ma da un altro valore a nostro piacere? Niente di più semplice! Basta infatti sommare il numero più basso della serie al risultato della funzione RND moltiplicato per l'intervallo esistente fra il minimo e il massimo. Ci affrettiamo a chiarire tutto con un semplice esempio; se volessimo generare solo numeri casuali compresi fra 50 e 80 procederemmo come segue:

```
A = 50 + RND(1)*(80-50 + 1)
```

Questa formula è sempre valida e si potrà rivelare molto utile in numerose occasioni. Notate che alla differenza dei due limiti deve essere aggiunto 1 altrimenti la funzione arriverebbe al massimo a 79. Fate però attenzione, perché in questo modo saranno generati numeri reali, quindi se siete interessati solo agli interi sarà bene che facciate precedere il tutto dalla funzione INT che abbiamo appena visto, riscrivendo il tutto come

segue:

A = INT (50 + RND(1)*(31))

Dunque 31 non è altro che la differenza fra 80 e 50 incrementata di 1 unità.
Oltre a queste funzioni di uso generale, il 64 dispone di una serie completa di funzioni trigonometriche e matematiche; queste sono: Sin() per il seno, COS() per il coseno, TAN() per la tangente, ATN() per l'arcotangente, LOG() per i logaritmi, EXP() per l'elevamento a potenza e SQR() per le radici quadrate.
La maggior parte di queste funzioni non richiedono particolari commenti, poiché si limitano a ricavare il valore di una determinata funzione matematica a partire dal valore che noi poniamo fra parentesi. L'unica che merita qualche precisazione è la funzione EXP(), la quale restituisce il valore dell'elevamento a potenza della costante 2.71828183, quindi l'istruzione

```
PRINT EXP (7)
```

equivale a scrivere

```
PRINT 2.71828182 T 7
```

Tutte queste funzioni non sono comunque di uso molto frequente e potrebbero interessare solo a chi intende usare il proprio computer per applicazioni matematiche.
Passiamo ora alle funzioni per il trattamento delle stringhe, che il Commodore può manipolare con relativa facilità.
La funzione ASC() fornisce come risultato il codice del primo carattere della stringa o variabile alfanumerica posta fra parentesi. Per conoscere i codici corrispondenti ai caratteri si consulti una tabella.
La funzione opposta a ASC() è CHR$(). Essa fornisce il carattere al codice numerico che si trova fra parentesi.
Una funzione molto utile è LEN(), essa infatti permette di conoscere la lunghezza della stringa in esame. Per esempio, se noi in un programma definiamo una variabile alfanumerica, possiamo poi sapere di quanti caratteri è composta applicando questa funzione. Vediamo un esempio:

```
10 LET A$ = "COMMODORE"
20 PRINT LEN(A$)
```

In questo modo otterremo la stampa di 9 che è appunto la lunghezza della stringa A$.
Veniamo ora al punto forte dell'elaborazione delle stringhe: l'estrazione e la manipolazione di parti costituenti una stringa principale. Le funzioni che ci permettono di trattare in modo completo le stringhe sono: LEFT$(), RIGHT$() e MID$().
La prima di queste funzioni consente di estrarre un certo numero di caratteri a partire dal primo a sinistra; se noi volessimo ricavare i primi 3 caratteri di sinistra della stringa a$ definita sopra procederemmo come segue:

```
PRINT LEFT$ (A$,3)
```

Ottenendo così la stampa di CON che sono proprio le prime 3 lettere della stringa COMMODORE sopra definita.

Se noi volessimo invece estrarre un determinato numero di caratteri a partire dalla destra della stringa dovremmo usare la funzione RIGHT$(), la cui sintassi è uguale alla precedente, quindi se scrivessimo:

PRINT RIGHT$ (A$,3)

Otterremmo la stampa di ORE che sono i 3 caratteri a destra della solita stringa.
La funzione MIO$() è un po' più complessa, in quanto permette di estrarre una sottostringa lunga un certo numero di caratteri a partire dal carattere specificato. Se noi volessimo quindi estrarre una stringa di 3 caratteri a partire dal 2°, scriveremmo:

PRINT MID$ (A$,2,3)

In questo modo sarebbe stampata la stringa OMM. Semplice vero?
Bene! Senza quasi rendercene conto abbiamo già affrontato le istruzioni BASIC del 64 e abbiamo esaminato le principali funzioni di cui esso dispone. Rimarrebbero da citare input/output e alcune funzioni particolari, ma di questo ci occuperemo un'altra volta, per il momento il consiglio migliore che possiamo darvi è di mettere subito in pratica tutto ciò che avete imparato. Buon lavoro.

Massimo Cellini

LE PERIFERICHE: DI INGRESSO E USCITA

Nel numero scorso questo articolo è risultato interrotto per un errore di stampa. Per lo stesso errore sono state stampate le istruzioni di 4 programmi (Formula 1, Geometria, Bosco Incantato, Esherland) non compresi nella cassetta di Play ON TAPE n. 5. Ci scusiamo e pubblichiamo di seguito la seconda parte dell'articolo sulle periferiche.

La verifica del codice vien fatta dal calcolatore e l'accordo garantisce entro certi limiti, l'esattezza del dato ricevuto.
Il numero di caratteri che è possibile registrare su un nastro di carta è, ovviamente, uguale al numero di combinazioni possibili delle perforazioni (256 nel caso di nastro ad 8 piste e 32 per uno a 5 piste).
La lettura di un nastro di carta può essere fatta solo in modo seriale, a causa della sua lunghezza non ben definita. Sono disponibili testine di lettura del tipo a spazzole e del tipo a fotocellule. Nel primo caso la velocità di lettura è di circa 180 caratteri al minuto, nel secondo può raggiungere i 1000 caratteri.
Anche la perforazione può essere fatta solo in modo seriale. La stazione di perforazione è simile a quella di un perforatore di schede, a parte la forma dei punzoni e la forza ad essi impressa: il nastro di carta è molto più leggero e meno resistente delle schede. Molto spesso il lettore e il perforatore sono accoppiati in un unica unità.

NASTRO MAGNETICO; Unità a nastro magnetico.

Il nastro magnetico che si usa per gli elaboratori di dati è simile a quello utilizzato dai comuni registratori: cambiano le dimensioni, la qualità, il metodo di registrazione. Analogamente a quanto si fa con il nastro di carta, le cifre binarie, che rappresentano un carattere, vengono registrate lungo la stessa riga trasversale del nastro, su otto piste diverse. La nona pista, come la riga check per il nastro di carta, ha la funzione di controllo di parità. (Nella nona riga viene registrato un 1, se il numero di 1 nelle altre piste è pari; uno zero, se il numero di uno nelle altre piste è dispari).
I dati vengono registrati su nastro magnetico uno di seguito all'altro e la fine di una registrazione è indicata da uno spazio vuoto, lungo circa 0.6 pollici. La funzione di questo spazio vuoto è quella di permettere al meccanismo di trascinamento del nastro di fermarsi a fine lettura e di riprendere la velocità di regime, prima della lettura suc-

cessiva.
Questo risulta necessario giacchè la lettura o la registrazione dei dati è fedele solo se il nastro si muove alla velocità standard prevista. L'insieme delle informazioni contenute in una registrazione è detto record: più precisamente, viene detto record fisico l'insieme di dati contenuti fra due spazi vuoti successivi.
Il calcolatore è strutturato per leggere record fisici e, al comando di lettura, trasferisce dal supporto in memoria centrale un intero record fisico.
Il nastro magnetico consente di leggere i records sequenzialmente, non è conveniente fare salti alla ricerca di una informazione, poichè i tempi di ricerca sul nastro sono molto alti. Generalmente si utilizzano nastri lunghi circa 700 metri e si registrano con una densità fino a 1600 caratteri per pollice. La velocitá di lettura/scrittura arriva sino a 300.000 caratteri per secondo.

DISCO MAGNETICO; Unità a disco magnetico.

Il disco magnetico è costituito da un piatto circolare di alluminio sulle cui facce è applicato uno strato di materiale magnetizzabile. Le piste sono circolari e concentriche, invece che a spirale continua come nei comuni dischi: ciò è dovuto al fatto che, mentre in questi ultimi è il solco a guidare il braccio, nei dischi del calcolatore il braccio deve potersi muovere liberamente, per assicurare l'accesso diretto alla pista o a parte di essa. Abbiamo detto, riguardo al nastro, che non consente di accedere direttamente a particolari informazioni ma è necessario trattarle sequenzialmente. Nel disco, pur essendo assicurati i vantaggi della registrazione magnetica, è possibile accedere ai dati direttamente (modo direct o random).

I bits in cui vengono codificati i caratteri sono registrati sequenzialmente sulla medesima pista. Il disco è tenuto costantemente in rotazione e l'accesso alle piste è realizzato spostando il braccio recante la testina trasversalmente. In realtà esistono due bracci distinti e solidali tra loro, uno per la faccia superiore e l'altro per la faccia inferiore del disco. Il meccanismo di spostamento del braccio è elettromagnetico o pneumatico. Lo spostamento massimo del braccio, cioè dalla pista più interna a quella più esterna, avviene in qualche decina di millesimi di secondo.

I sistemi di elaborazione di grandi dimensioni utilizzano unità a dischi con più piatti sovrapposti. L'insieme delle piste corrispondenti sui vari dischi viene chiamata cilindro e l'insieme rigido dei piatti è detto DISK-PACK.
La velocità di rotazione del disco è di circa 4000 giri al minuto. La velocità di lettura/scrittura si aggira sui 300.000 caratteri per secondo.

ISTRUZIONI PER LA CASSETTA DI PLAY ON TAPE COMPUTER N. 6

ISTRUZIONI DI CARICAMENTO

Prendete la cassetta, mettetela nel registratore con la facciata prescelta, A oppure B, in alto. Accendere poi il computer e premere "SHIFT" e "RUN STOP" contemporaneamente: apparirà la scritta "Press Play On Tape" che significa "Schiacciate il tasto Play sul registratore".

Eseguite il comando e attendere cha appaia la scritta 'FOUND INTRODUZIONE A', oppure, se state caricando l'altra facciata, "B". A questo punto premete la barra spaziatrice e attendete qualche secondo fino a che lo schermo riprende a mostrare il cursore lampeggiante. Se dovesse apparire qualche scritta di errore, riavvolgete la cassetta e ripetete le operazioni. Se invece tutto è ok, schiacciate il tasto "STOP" sul registratore: dopo qualche istante apparirà la prima pagina della nostra rivista con l'elenco dei programmi contenuti nella cassetta. Quando vedrete la scritta "Premere il tasto" schiacciate un tasto a caso sulla tastiera.

Non appena lo schermo sarà ritornato blu con il borzo azzurro e con le scritte nella parte superiore che normalmente vedete quando accendete il computer, premete nuovamente "SHIFT" e "RUN STOP"; schiacciate quindi il tasto "Play" sul registratore, attendete che il computer trovi il programma successivo, premete la barra spaziatrice e pazientate un attimo finché ricomparirà il cursore lampeggiante. Se non appaiono scritte di errore, premete STOP sul registratore.

Seguite queste semplici istruzioni ogni qualvolta volete caricare un programma di Play ON TAPE Computer.

Qualora trovaste difficile caricare i programmi e apparissero sullo schermo le scritte di errore (es. "PRINT LOAD ERROR") potreste tentare di modificare leggermente la posizione delle testine con un piccolo cacciavite inserito nel foro posto nella parte superiore del registratore eseguire 1/2 giro a destra o sinistra. Dopo qualche tentativo non dovreste avere più problemi.

Se volete cambiare programma, avete due alternative: A) premete il tasto "RUN/STOP" (per il C. 16 il tasto di RESET); se nulla accade, sempre tenendo premuto, battete una o più volte il tasto "RESTORE": lo schermo dovrebbe ripulirsi e apparire blu con il

bordo azzurro. A questo punto seguite le solite istruzioni di caricamento; B) se proprio non riuscite ad uscire da un programma, niente paura: spegnete e riaccendete il computer e ricominciate seguendo le consuete istruzioni per caricare i programmi.

(LATO A: VIC 20)

REDKNIGHT

Entra nel mondo magico e avventuroso del Medioevo con questo "Adventure Game" (in Italiano).

Nell'Adventure, come certamente saprai, non bisogna "smanettare" col Joystick o con la Tastiera, per risolvere le varie situazioni di gioco, ma dare delle istruzioni scritte al computer. Se ad esempio il computer ti dice: SEI IN UNA STANZA. Tu gli puoi rispondere: OSSERVO.

Se c'è qualcosa da vedere il computer te lo dice: SEI IN UNA STANZA E CI SONO DELLE FRECCE.

La tua risposta può essere: PRENDO FRECCE.

Per spostarti da un luogo all'altro, invece, le indicazioni sono semplici: N = NORD, S = SUD, E = EST, O = OVEST.

In questo Adventure sei REDKNIGHT (Cavaliere Rosso) e vuoi sposare la Principessa di Viclandia.

Ma la figlia di un Re, nel Medioevo, non si sposa così facilmente. Devi prima superare Tre dure prove per dimostrare al tuo futuro Suocero di essere un suo degno successore al Trono. Nella prima prova devi scoprire il nome del Re, che solo una persona conosce. Scopri, perciò, chi sia costui e arriva al suo cospetto attraverso varie peripezie, che però non sono casuali: ogni cosa che incontri, ogni azione che fai crea i presupposti per questo incontro.

Perciò trovandoti davanti a qualcuno se alla domanda CHIEDO NOME, non hai alcuna risposta vuol dire che non è la persona giusta; se, invece, hai la risposta: NON POSSO DIRTELO, la persona è giusta, ma non hai fatto, durante il gioco, qualcosa che gli permetta di dirtelo.

Quando hai scoperto il nome torna dal Re e ....

Nella seconda avventura devi porre fine alle scorribande del Drago Alato e tornare dal Re.

Affronta, ora, nella terza impresa, il Folle Mago, uccidilo e torna dal Re.

Un consiglio: per meglio giocare un Adventure prendi carta e penna e scriviti lo schema del percorso che stai facendo.

Così, dopo una morte, ricominciando, non correrai lo stesso rischio.

Ti dò ora un Piccolo Glossario delle parole che il computer accetta. Alcune di queste sono valide per tutte e tre le avventure (es. Osservo-Os), altre solo in una o due.

Mancano, naturalmente delle parole che dovrai scoprire du-

rante il gioco e che servono per la soluzione.

Apro, brucio, incendio, alzo, aiuto, offro, osservo, os, mago, me, N, S, E, O, prendo, chiedo, masso, drago, grotta, fiori, dico, scrivo, do, dono, sollevo, esco, cibo, infilzo, mangio, sughero, chiave, torcia, sussurro, mormoro, cuoco, caverna, talismano, re, nome, attraverso, passo, urlo, prendo, raccolgo, cascata, foglie, scendo, trafiggo, insetti, spada, rupe, porta, savana.

Più di così non posso dirti.

Parti lancia in resta e sposati.

RISPARMIATORE

Com'è difficile riuscire a mettere pochi soldi nel salvadanaio al giorno d'oggi!!

Fra inflazione galoppante ed esattori c'è poco da stare allegri.

Guida, con la tastiera, la tua moneta e cerca di infilarla nel salvadanaio.

I TASTI CONTROLLO SONO:
la prima fila di tasti (1-2-3 ecc.) = SU
la seconda fila di tasti (Q-W-E-R ecc.) = SINISTRA
la terza fila di tasti (A-S-D-F ecc.) = DESTRA

Il valore della moneta, causa l'inflazione, è decrescente: parte dal massimo e arriva a Ø, col passare del tempo.

Se lasci svalutare la tua moneta fino allo Ø hai perso parte del tuo capitale.

Stai attento anche a non urtare gli esattori in agguato o a non fare cadere la moneta fuori dal salvadanaio.

Ogni 3 monete perse finisce il gioco.

QUINDICI 3D

Quando ancora il Cubo di Rubik e gli altri "Rompicapi" suoi derivati erano di là da venire, il Gioco del 15 imperava.

La meccanica del gioco era, ed è, molto semplice: riordinare, in ordine crescente 15 cubi numerati dall'1 al 15, con l'1 posto nell'angolo superiore sinistro, spostandoli uno alla volta, in senso orizzontale o verticale.

La possibilità di movimento ti è data dal "buco" libero che, all'inizio del gioco, è nell'angolo inferiore destro.

Alla destra dei cubi da riordinare, un orologio scandisce il tempo e un contatore ti dice quante mosse hai fatto.

I tasti controllo sono: ϱ = su, = = destra
/ = giù, : = sinistra

La sfida è aperta anche ai "Cubisti", che forse tendono a sottovalutare questo splendido gioco.

API E RAGNI

Quando mangi una fetta di pane con burro e miele, forse, non pensi a quanto sia difficile per le "buone" Api raccogliere il nettare dai fiori.

Anche i Ragni hanno fame e le Api sono, per loro, un buon pranzo.

Vola da un fiore all'altro, ma ricorda che, quando ti posi la prima volta su uno di essi, svegli il Ragno che ci dorme dentro.

Un secondo passaggio su un fiore già "succhiato" ti trasforma nel pranzo del Ragno svegliato. (Ho fatto la rima senza pensarci prima).

MORALE: la vita non è "nettare e miele" neanche per le Api.

Tasti Controllo: ϱ = su, : = sinistra
/ = giù, = = destra

OROSCOPO

Chi, anche fra i più scettici, non ha letto almeno una volta il proprio Oroscopo?

I tempi dell'Oroscopo "generico", letto sul giornale, sono finiti. Ora col tuo Computer puoi farti un tuo Oroscopo personale per quanto riguarda la Salute, l'Amore, il Lavoro.

Inserisci la data di nascita.

Gi = Giorno (in cifre) scrivi i tuo giorno di nascita e premi RETURN.

Me = Mese (in cifre) inserisci il mese di nascita e premi RETURN

An = Anno (in cifre) inserisci l'anno di nascita (tutto intero es. 1953) e premi RETURN.

Inserisci, nello stesso modo, la data del giorno.

Scrivi ora il tuo nome e cognome, senza lasciare spazi fra i due, e premi RETURN.

Premi ora il tasto 1 se il tuo umore è Buono

Premi ora il testo 2 se il tuo umore è Normale

Premi ora il tasto 3 se il tuo umore è Cattivo

Il computer ti mostra ora il tuo Segno Zodiacale; premi un tasto e premi il Play sul Registratore.

Si carica ora il programma che, elaborati i dati inseriti, ti dà il responso.

BIORITMO

Questo è un programma che, come dice il nome stesso, vi permette di conoscere il vostro bioritmo e magari quello di amici o dei giocatori della vostra squadra preferita.

Sarà sufficiente digitare, dietro precisa domanda del computer, il nome di battesimo, la data di nascita e quella a partire dalla quale desiderate conoscere il vostro stato futuro ed il computer in breve elaborerà i dati fornitigli e li tradurrà in un chiaro grafico dove saranno visualizzate le linee riguardanti lo stato fisico, quello emotivo e quello intellettuale durante i 34 giorni successivi alla data di partenza da voi fissata; è poi possibile visualizzare un secondo gruppo di 34 giorni, e quindi un terzo e così via; oppure potete inserire nuovi dati riguardanti una seconda persona.

Infine, se siete fortunati possessori di una stampante, avete la possibilità di ottenere una copia su carta del bioritmo desiderato.

ROULETTE

Le regole sono quelle normalmente in vigore ai tavoli della roulette (tranne qualche limitazione nei tipi di puntata) e per altro vi verranno esaurientemente spiegate nel corso del gioco stesso. Varrà qui la pena di ricordare solo il sistema di puntate accettato dal computer, al fine di averlo sempre davanti agli occhi nel corso della partita e di evitare così qualsiasi errore:

R = Puntata sul rosso
N = Puntata sul nero
P = Puntata sul pari
I = Puntata sul dispari
M = Puntata sul manque
PA = Puntata sul passe
S(Ø/36) = Puntata sul numero singolo (da Ø a 36)
C(1/3) = Puntata sulla colonna (prima, seconda o terza)

Informate il vostro Commodore 16 circa il il numero di partecipanti al gioco, la loro posta globale e l'entità delle singole puntate; esso, oltre che un onestissimo croupier, si rivelerà un prezioso segretario nel tenere l'esatto computo delle somme vinte o perse da ciascun giocatore. Buona fortuna!

TOTOCALCIO

Il programma permette di scrivere una colonna pronostico della settimana fornendo alcuni dati sulle squadre in schedina. È possibile elaborare la colonna con una dose più o meno ampia di casualità. È noto a tutti che esistono risultati molto o poco scontati; se l'indice di casualità è basso, il computer fornirà l'elenco di previsioni basandosi soprattutto sui dati inseriti (classifica, forma ecc...); ma non sempre i risultati scontati si verificano ed è proprio allora che si verificano le vincite più consistenti.

La casualità, da inserire prima di elencare le squadre, è dunque il fattore che rende la colonna elaborata più o meno ricca di sorprese. I valori vanno da 5 a 25, con 5 = poca casualità, 25 = molta. Occorre inserire l'indice sia per le squadre in casa che in trasferta: dipende dal giudizio che ognuno di noi può dare sulla possibilità di sorprese fornite dalle squadre ospitanti o ospiti. Dopo aver scritto i valori di casualità e aver battuto 'RETURN', se si è commesso un errore, basterà premere 'C' e ripetere correttamente l'operazione. Per proseguire occorrerà poi battere un tasto a scelta. Il computer chiederà il nome delle squadre in schedina. Si comincia dalla prima in alto a sinistra e si proseguirà fino all'ultima in basso a destra. Occorrerà scrivere le squadre che si scontrano di fila, una dopo l'altra, scrivendo prima la squadra ospitante, poi quella ospite. Di ognuna vengono richiesti: i punti in classifica e lo stato psicofisico, cioè la forma e il morale, (da 0 a 10). Dopo ogni dato bisogna premere 'RETURN'. In caso di errore basta rispondere alla domanda 'SQUADRA?' con 'CO' e premere 'RETURN', indi ripetere correttamente l'operazione. Le squadre da inserire sono 26, infatti le partite in programma e quindi i pronostici sono 13. Inserita l'ultima squadra, il programma fornirà uno schema riassuntivo delle prime tredici squadre con i punti e la situazione psicofisica; premendo un tasto si ottiene la stessa cosa per le ultime 13.

WARGAME

Wargame è una simulazione di guerra tra le 2 superpotenze USA e URSS. Si gioca in 2 concorrenti ognuno dei quali impersona una delle 2 nazioni. Una volta caricato il programma, dopo aver battuto il 'RUN', il gioco richiede di scegliere tra una partita a tempo o a limite di punti. Nel primo caso il vincitore sarà colui che allo scadere del tempo di gioco si troverà in vantaggio, nella seconda vincerà colui che raggiunge il punteggio stabilito. Per scegliere il tipo di partita basta premere 'T' per tempo, 'P' per punti. Il tempo di gioco va scritto nella forma 'HHMMSS', ovverosia ore, minuti, secondi. Qualche esempio: 2 ore, 10 minuti e 20secondi andrà scritto 021020; 12 ore, 3 minuti e 0 secondi 120300; 25 minuti e 50 secondi 002550. Battute le cifre premere 'RETURN'. Nel caso si preferisca una partita con un obbiettivo di punti, dopo aver premuta la 'P', scrivere il punteggio d'arrivo e

battere 'RETURN'.

Ora ha inizio il gioco. Lo schermo ci presenta a sinistra la pianta del mondo con sotto la scritta 'ATTENDO ORDINI', mentre a destra sono evidenziati il punteggio e la situazione bellica dei 2 contendenti. Nella parte inferiore dello schermo sono invece elencate le città obbiettivo della nazione cui tocca il turno. Gli ordini che il computer attende sono di 3 tipi e per ricordarli basta premere il tasto funzione 'F7': 'SPOSTAMENTO TRUPPE' (F1), con il quale è possibile ottenere vantaggi strategici che permettono di conseguire risultati migliori in caso di combattimento: 'ATTACCO ATOMICO' (F3), con il quale è possibile distruggere una delle città nemiche elencate nella parte inferiore dello schermo: 'DIVERSIVO POLITICO' (F5) con il quale, per mezzo della diplomazia, si possono ottenere punteggi senza combattere. Analizziamo più approfonditamente i 3 comandi. Per operare uno spostamento di truppe basta premere il tasto funzione 'F1' (2 volte se si era richiesto l'aiuto con il tasto 'F7'); lo schermo ci fornisce il numero e la dislocazione delle divisioni delle 2 superpotenze in modo da poter controbilanciare eventuali svantaggi strategici o per ottenerne. Accanto alle truppe della nazione cui tocca il turno sono elencati i 3 tasti funzioni con cui si spostano gli eserciti; il computer richiede la provenienza, indi la destinazione e infine il numero (il massimo sono le divisioni presenti nel territorio di provenienza); scritto il numero battere 'RETURN' e premere un tasto. Per attivare l''ATTACCO ATOMICO' è necessario premere 'F3' (2 volte se si era chiesto l'aiuto con 'F7'). Lo schermo indica la potenza delle testate nucleari rimaste e richiede il bersaglio, cioè una delle città sottoelencate; scritta la città, premere 'RETURN' e scrivete la potenza scelta. Si ricorda che vi è un limite massimo alla potenza utilizzabile, frazione delle testate nucleari possedute. Battuto il 'RETURN' il turno passa all'avversario, ma con l'attacco atomico si ha distrutto una città (che apparirà d'ora in poi accompagnata da un asterisco), ucciso dei civili e ottenuto un punteggio in classifica.

Il 'DIVERSIVO POLITICO' è invece un raffinato strumento diplomatico. Dopo averlo selezionato con 'F5', si ottiene il servizio CIA o KGB, a seconda del concorrente in gioco. Il servizio riferisce i punti diplomatici dei 2 Paesi (da non confondere con i punti in classifica) e 2 alternative: 'DISTENSIONE' o 'SOVVENZIONI', selezionabili attraverso i tasti funzione. Con la 'DISTENSIONE' (F1) è possibile distruggere parte del proprio arsenale militare ottenendo così punti diplomatici che, se non bilanciati dall'avversario, porteranno in seguito punti nella classifica generale. Con le 'SOVVENZIONI' (F3) si possono ottenere direttamente punti nella classifica generale: distribuendo a paesi terzi punti diplomatici ottenuti con la 'DISTENSIONE', si costringe l'avversario alla trattativa (il turno passa la contendente con scelta automatica del 'DIVERSIVO POLITICO') e la differenza tra le sovvenzioni distribuite dai 2 paesi fornirà punti in classifica generale a colui che ne avrà fatte di più. A voi dunque le armi e vinca il migliore incrociando le dita: meglio che una simile guerra sia un gioco, piuttosto che una realtà.

COLOUR SEARCH

Vi sono 7 colori a disposizione; il computer dispone di 5 caselle rappresentate da 5 'Q'. Esso sceglierà una combinazione segreta di tonalità (es: rosso, verde, giallo, rosso, blu) ponendo in ogni casella un colore, potendo anche ripetere i colori usati. Il giocatore deve tentare di indovinare la combinazione segreta scegliendo tra i 7 a disposizione. In basso sullo schermo sono ricordati i colori utilizzabili. Per scegliere un colore basta premerne l'iniziale ('B' sta per blu, 'P' per porpora, 'V' per verde ecc...). Poiché è molto difficile indovinare al primo colpo, per aiutare nella ricerca il calcolatore fornisce una mano al giocatore. Per ogni colore indovinato ma collocato nella casella sbagliata darà una 'Z' chiara, per ogni colore esatto nella casella esatta una 'Z' scura. Una volta indovinata la combinazione segreta, dopo i complimenti di rito, se il numero di tentativi utilizzati è sufficientemente basso, si potrà scrivere il proprio nome per entrare nella classifica (hall of fame) mostrata all'inizio del gioco. Dopo aver scritto il nome, battere 'RETURN'. Buona ricerca.

PLAY ON TAPE COMPUTER 6

VIC 20

COMMODORE 16

VIC 20:

- REDKNIGHT
- IL RISPARMIATORE
- API E RAGNI
- QUINDICI 3D
- OROSCOPO

COMMODORE 16:

- TOTOCALCIO
- COLOUR SEARCH
- ROULETTE
- WAR GAME
- BIORITMO

EV EDITORIALE VIDEO