PLAY ON TAPE

COMPUTER

manuale 8

VIC 20 · C.16 PLUS/4

LINGUAGGIO: MACCHINA
Le istruzioni di caricamento

PSEUDOCODICE: I FILES

GLI ALGORITMI FONDAMENTALI

EV EDITORIALE VIDEO

Il club ideale degli utilizzatori di Vic 20 e C. 16 potrebbe costituire un partito, e forse non dei più piccoli. Lo staff di PLAY ON TAPE Computer fa parte di questo partito e ne è entusiasta, riceve idee e consigli dai lettori e li attua con gioia, dando un senso sempre più entusiasmante alla propria attività di ogni giorno.

Gli argomenti di questo numero sono, come sempre, il risultato dell'entusiasmo e dell'affiatamento tra redazione e lettori. La puntata di Pseudocodice questa volta si occupa dei Files chiarendone il concetto e approfondendolo al tempo stesso. Ed è subito Basic, classico e importante: Come tradurre un numero esadecimale alla base 2 o 16" è il tema dell'articolo seguente.

La lezione di linguaggio/macchina di questa puntata considera alcune istruzioni importantissime per i futuri programmatori in L/M: le istruzioni di caricamento e il trasferimento dei dati. Ed è ancora Basic: cerchiamo di conoscere gli Algoritmi fondamentali.

La cassetta, dalla parte del Vic 20, presenta Seven Eleven, una bellissima versione del classico gioco; Meteore, che mette alla prova la vostra abilità; Telefono, programma che vi consente di calcolare sempre il prezzo delle vostre telefonate; The last battle, battaglia intergalattica all'ultimo missile; infine il Riflessometro, un gioco-programma che consente di misurare divertendo i propri riflessi e indirettamente anche lo stato della forma.

Dal lato del Commodore 16 troviamo un'altra versione del Riflessometro; poi Quadrati, gioco di strategia per misurare la propria abilità in una sfida di astuzia e calcolo; il Contatore Telefonico, sempre per calcolare i costi delle spese telefoniche; C. 16 dà un valore alla nostra memoria visiva e allo spirito di osservazione; Dama, infine, è una fantastica versione del conosciutissimo gioco di abilità: attenzione, il computer è un avversario formidabile.

Questi i contenuti del numero 8: il prossimo conterrà nuove ed entusiasmanti sorprese.

Arrivederci a presto.

Editoriale VIDEO - Direttore: Antonio Lucarella - Coordinamento tecnico: Roberto Treppiedi - Hanno collaborato: Franco Longoni, Massimo Cellini, Fabrizio Iotti, Maria Russino, Aldo Campanozzi, Silvana Buttà, Cinzia Agiman, Domenico Cellamare, Roberta Di Pietro, Alessandro Vallone, Arnaldo Restelli, Dino Ticli, Igino Zaffaina, Giovanna Zampella - In copertina: disegno di Aldo Campanozzi - Stampa: Color Graf Milano - Fotocomposizione: ERREGI Milano - reg. del Trib. di Milano n. 518 del 10/11/84 - Stampato in Italia

I FILES

In questo articolo ci occuperemo del concetto di FILE (pronuncia: "fail") e delle frasi di pseudocodice che ci permettono di lavorare con i files, cioè le frasi "Leggi file" e "Scrivi file"; vedremo inoltre alcuni algoritmi in pseudocodice che racchiudono le funzioni base della scansione, analisi, caricamento e aggiornamento di un file.
Per introdurre il concetto di file, diremo subito che in inglese questo termine significa "ARCHIVIO" costituito da schede; la cassettiera con gli scompartimenti che segnano l'inizio di un blocco di schede, normalmente in ordine alfabetico, è chiamata in inglese "file cabinet"; ogni scheda ("file card") contiene una registrazione ("record").
Il significato inglese di file e record, cioè letteralmente "archivio e registrazione", è quindi un significato generico, associato alla vita di ufficio, fin da prima dell'introduzione del computer.
In italiano le parole file e record non vengono tradotte, e hanno solamente il significato tecnico acquisito con l'introduzione del computer. È però esemplificativo mantenere l'analogia con la realtà concreta di un archivio cartaceo, per comprendere bene come funzionano le cose, in modo da farcene una immagine mentale appropriata.

IL CONCETTO DI FILE

Un file è una raccolta di records, memorizzati sulla superficie magnetica di un nastro o di un disco.
Ogni record è la registrazione di dati (valori) relativi ad un singolo avvenimento: per esempio, se intendessimo memorizzare un'agenda telefonica, ogni record conterrebbe il nome, cognome, indirizzo e numero telefonico di un singolo personaggio.
L'analogia con il file cartaceo è evidente:

- il nastro magnetico (oppure il dischetto) è il "file cabinet",
- il singolo record del file è la registrazione di una "file card" del nostro archivio, dove la "scheda" è quella piccolissima porzione di nastro che contiene i dati di un unico record.

Come si vede, il concetto di scheda singola è un po' evanescente: non è come avere tanti cartoncini separati e ben visibili a nostra disposizione, ma dobbiamo lavorare di fantasia per comprendere che ogni record è in effetti una registrazio-

ne separata per il nostro personal.
Proseguendo nell'analisi, notiamo che un record contiene un numero fisso di informazioni, chiamate "campi" del record. Tornando all'esempio dell'elenco telefonico, il nome, cognome, indirizzo e numero di telefono sono i 4 campi del record. I campi del record si possono memorizzare uno di seguito all'altro, basta che noi conosciamo il preciso significato di ognuno di essi: sono posizionali e perciò in un record non è necessario memorizzare altro oltre ai valori stessi dei vari campi. La descrizione dei campi in ogni record di un certo file si chiama tracciato record. Per il file "Elenco Telefonico" avremmo il seguente tracciato record:

Nome	Cognome	Indirizzo	N° Telefono

La lunghezza di ogni campo è (in genere) fissa: per esempio 10 caratteri per il nome, 25 per il cognome, 30 per l'indirizzo e 12 per il numero di telefono. Di conseguenza, ogni record contiene un numero fisso di caratteri, (nel nostro caso 77).
In ogni linguaggio di programmazione, esiste una istruzione primitiva che scrive un record sul nastro o sul dischetto, ed un'altra che legge un record dal nastro alla memoria del personal. L'informazione passa infatti dalla forma di elettricità della memoria RAM (Random Access Memory; memoria ad acceso casuale) alla forma di magnetismo indotto su una minuscola area di materiale magnetizzabile depositato sulla superficie del nastro o del disco, mentre questo scorre sotto una testina di lettura/scrittura.
La grossa differenza tra le due forme di memorizzazione delle informazioni, è che nel caso della RAM il contenuto di informazione viene "perduto" non appena cessa l'alimentazione di elettricità (cioè quando si "spegne" il personal), mentre sul nastro o disco, il magnetismo indotto rimane "per sempre".
Perciò i dati dei programmi, che debbono essere conservati come patrimonio acquisito nel tempo, debbono essere mantenuti su nastro o disco. Perciò il file rappresenta il mezzo per memorizzare informazioni su un supporto magnetico.

DIVERSI TIPI DI FILE

Dal punto di vista della programmazione la gestione di un file si riduce a queste 4 azioni semplici:

1. raccogliere le informazioni che compongono un record del file (da tastiera)
2. scrivere il record su supporto magnetico in un certo file
3. leggere un record da supporto magnetico da un certo file
4. separare le informazioni che compongono un record nelle componenti singole.

Le azioni 1 e 2 portano a termi-

ne una **scrittura** di un record in un file, e le azioni 3 e 4 una **lettura**.
Sotto il controllo di un programma le informazioni in RAM sono contenute in variabili, e perciò le azioni 1 e 4 consistono rispettivamente nel concatenare diverse variabili in un'unica variabile, e scomporre la variabile che contiene il record letto in molte variabili. Conviene, per ragioni di stile, mantenere nei files sempre valori alfanumerici in modo che le operazioni sopra descritte siano la semplice concatenazione (simbolo " + "), e la estrazione che in pseudocodice si chiama "estrai" (ed in BASIC si traduce con la MID$).
Predisporremo quindi nel programma una variabile di nome REG che servirà come variabile di passaggio per la scrittura e lettura del file.
Rimane da chiarire ora un punto molto importante, che potremo comprendere meglio con un esempio.
Nel caso di un file di schede di carta, una operazione di scrittura consiste nel completare una scheda bianca. A questo punto la introduzione vera e propria della scheda (record) nell'archivio (file) è una operazione successiva e che abbisogna di una ulteriore informazione: in quale punto dell'archivio devo inserire la nuova scheda?
La risposta potrebbe essere:
1. metti via la scheda dopo tutte le altre già nell'archivio
2. mettila nella posizione N-sima
3. mettila nella sequenza alfabetica, rispetto ad un campo chiave (per esempio il nome)

A seconda della nostra scelta, abbiamo 3 tipi di archivio:
1 sequenziale
2 random (o meglio "diretto per numero")
3 diretto per chiave.

I tre tipi di archivio sono sempre più sofisticati andando da 1 a 3; sul personal computer esistono solo i tipi 1 e 2 come struttura file elementari.
Il file sequenziale, cioè quello in cui ogni nuovo record viene aggiunto in coda a quelli già esistenti, è il più semplice, e può essere realizzato anche su nastro. Il file random non può essere realizzato su nastro ma solo su dischetto perché una scrittura o lettura comporterebbe uno scorrere avanti o indietro del nastro per potersi posizionare sul punto giusto che non sarebbe accettabile nè del punto di vista del tempo nè dalla precisione della meccanica del nastro.

I FILES SEQUENZIALI

Trattiamo ora in dettaglio il file sequenziale, prendendo come esempio l'elenco telefonico dei nostri amici.
Diciamo subito che ogni file deve essere dichiarato e "aperto" prima di iniziare a lavorarci, e deve essere "chiuso" a lavoro ultimato. Nell'operazione di apertura ("open") si specifica se il file esiste già, se si intende creare da zero, o se si desidera

solamente leggerlo. In BASIC si indicano tutte queste cose, oltre ad un numero di identificazione del file; in pseudocodice basta scrivere la frase: "Apri file" seguita dal nome del file.
Vediamo come si presenta un algoritmo per la creazione del file Elenco, ottenuto leggendo da tastiera i contenuti dei vari campi (fino a che si scrive "***") e scrivendo un record alla volta nel file:

```
INIZIO. Costruzione
(e ampliamento)
Elenco Telefonico
  Apri file "Elenco"
  RIPETI
  FINCHÈ (CTL = "***")
    Ricevi da tastiera nome,
    cognome, numero telefonico
    Componi RECORD = nome
     + cognome + num.telef.
    Scrivi RECORD nel file
    "ELENCO"
  FINE_RIPETI
  Chiudi file "Elenco"
FINE
```

Come si vede, l'algoritmo è molto semplice: la variabile RECORD serve da trainante per comporre le informazioni e poi scriverle nel file.
La figura 1, mostra come in BASIC si realizza questo algoritmo. L'unica cosa da notare è che si è usata l'accortezza di "inscatolare" ogni campo in una stringa di lunghezza fissa, in modo che il nostro file abbia i campi ben distinti in zone: questo si ottiene aggiungendo ad ogni variabile (COG$ per esempio) una stringa di spazi (52 per COG$) tale che sommata alla lunghezza della variabile si abbia una stringa di lunghezza costante.
Terminata questa fase il nostro file è memorizzato. Vogliamo vedere cosa contiene? Ci serve un algoritmo di "scansione" di un file sequenziale. Si tratta di aprire di nuovo il file - in sola lettura questa volta - e di scorrerlo dal primo all'ultimo record, stampandone i records a mano a mano che li leggiamo.
La SCANSIONE del file sequenziale è una ripetizione di lettura da un file condizionata dalla fine dei records da leggere. Quest'ultima condizione si chiama "fine del file" (in inglese End Of File abbreviato con EOF). Ci troviamo di fronte alla necessità di scrivere in pseudocodice una ripetizione condizionata da un evento esterno, l'EOF, che si rileva all'atto della lettura dell'ultimo record.
Abbiamo quindi una prima versione della scansione file:

```
INIZIO. Scansione file
sequenziale.
  Apri file in INPUT
  Leggi record in REC
  RIPETI
  FINCHÈ EOF
    Estrai campi da REC
    "Elabora campi"
    Leggi record in REC
  FINE_RIPETI
  Chiudi file
FINE
```

Questo famoso algoritmo, che si chiama "Scansione di file sequenziale con lettura fuori giro"

rappresenta il modello di scansione di qualunque file sequenziale, dove è sufficiente sostituire la giusta gestione di EOF ed espandere la parte "Elabora campi" per ottenere qualunque elaborazione per scansione di un file sequenziale.
Infatti, a seconda del personal, la gestione dell'EOF differisce nei particolari, ma il concetto resta il medesimo. Si tratta di leggere sul manuale quale indicazione di EOF è prevista dal BASIC installato.
Una variante di questo algoritmo fa uso di una struttura di ripetizione ancora non vista: la "Ripeti Per Sempre". Questa struttura, che rappresenta la ripetizione incondizionata, non va mai usata da sola, ma sempre in compagnia di un SE particolare. Ecco come si scrive una struttura di RIPETI PER SEMPRE:

```
INIZIO. Esempio di RIPETI
PER SEMPRE.
 RIPETI
 PER SEMPRE
  Az. A
  Az. B
   :
   :
  SE <condizione>
  ALLORA ESCI
 FINE_RIPETI
FINE.
```

Il ciclo "Ripeti per sempre" non terminerebbe mai; il SE...ALLORA ESCI funziona da controllore: non appena la condizione è vera, il ciclo si arresta.
Il vantaggio della ripetizione per sempre ("forever" in inglese) è che pone l'accento sul fatto che la condizione di interruzione non è logicamente contenuta nel programma - come nel caso del RIPETI FINCHÈ - e neppure dipende da un conteggio - come nel RIPETI PER I DA 1 A N -, ma dipende da un FATTO ESTERNO, nel nostro caso dalla fine del file.
L'algoritmo di scansione, in questo caso diventa:

```
INIZIO. Scansione del file
sequenziale ELENCO.
 Apri "Elenco" in INPUT
 RIPETI PER SEMPRE
  Leggi un record
  Stampa un record
  SE EOF ALLORA ESCI
 FINE_RIPETI
 Chiudi "Elenco"
FINE.
```

In BASIC la traduzione del "ripeti per sempre" si realizza così:

```
 10 "Ripeti per sempre"
 20 Az. 1
 30 Az. 2
  :  :
  :  :
100 IF<condizione> GOTO
      120
110 GOTO 10
120 "Fine Ripeti"
```

Un esempio di scansione e stampa del file Elenco seguendo l'algoritmo del "Ripeti per sempre" si trova in figura 2.
Eseguendo il programma in BASIC presentato nelle figure 1 e 2 alcune volte, vi accorgerete che:
1. i record nuovi vengono scritti

in coda a quelli già esistenti nello stesso file
2. l'ordine di lettura dei record è il medesimo dell'ordine in cui sono stati introdotti.

Questo è appunto ciò che significa operativamente il file di tipo SEQUENZIALE. Il che significa che le operazioni più normali fatte con un file sequenziale riguardano la scansione completa dell'intero file. Per esempio:
"Stampa l'elenco di tutti i numeri telefonici memorizzati", oppure "Dimmi quanti numeri di telefono posseggo". Memorizzando per esempio tutte le operazioni di incasso e di spesa in un file sequenziale possiamo avere l'elenco di tutte le spese con il loro totale, di tutti gli incassi ed il loro totale e la differenza tra i due.

I FILES AD ACCESSO DIRETTO

Ciò che non si può fare con un file sequenziale è scrivere un programma che risponda direttamente alla domanda: "che numero di telefono ha il mio amico Bianchi Mario?"
L'unica soluzione (improponibile) è quella di aprire il file, scandirlo tutto fino a trovare il record di Bianchi Mario, chiuderlo e ricominciare tutto daccapo per la prossima richiesta. Questo tipo di soluzione è tecnicamente da scartare, anche se pensiamo che varrebbe la pena di provarla sul serio per poter rilevare di persona dove stanno le controindicazioni.
La risposta tecnicamente valida al problema del recupero di un singolo record sulla base dei valori contenuti in un campo, è di costruire insieme al file dei dati, un qualcosa che permetta di raggiungere un record direttamente. Bisogna anche cambiare tipo di supporto fisico: dal nastro passare al dischetto.
Nel caso del file "diretto per numero di record" (il numero 2 dell'elenco di prima), bisogna poter dire qual è il numero di sequenza del record che contiene l'informazione. Nel caso della domanda posta in precedenza, si tratta di sapere che numero ha il record che contiene i dati di Bianchi Mario. Questo si può ottenere costruendo sia in lettura che in scrittura un numero ottenuto dal nome (magari sommando un valore per ogni lettera del cognome), e usando quel numero per memorizzare o recuperare il record. Questa tecnica si chiama "hashing".
Un'alternativa è memorizzare una tabella (che poi verrà memorizzata in un file sequenziale, come vedremo tra poco) che contiene i nomi ed i numeri di records del file dati. Questa tecnica si chiama "indicizzazione".
Nel caso in cui l'indicizzazione sia gestita in modo automatico (non è il caso del BASIC), il file assume la struttura di un file ad indice, e l'accesso si può fare indicando solo il campo chiave (files ad "accesso diretto per chiave", il numero 3 della lista di prima).

Notiamo che a mano a mano che il tipo di file diventa più complesso, bisogna trovare algoritmi sempre più sofisticati per gestire scansioni, aggiunte ed eliminazioni di records dal file; per questo consigliamo al lettore interessato di provare prima a gestire correttamente i files sequenziali, e poi di cimentarsi nella costruzione di files ad accessi più complicati. La letteratura sull'argomento è facile da trovare e sempre piena di buoni spunti.

FILE E TABELLA

La struttura di file è una struttura dati ESTERNA, perché risiede su supporto magnetico. Con la struttura dati "file sequenziale" abbiamo visto che ci è impossibile rispondere a domande dirette (del tipo "Che numero di telefono ha BIANCHI MARIO?") andando a cercare su disco il record che ci interessa. Però abbiamo una grossa possibilità: trasformare la struttura dati esterna file nella struttura dati interna tabella. In ciò consiste l'operazione inversa alla memorizzazione di un file su disco che si chiama caricamento di un file in TABELLA.
Le strutture file e tabella hanno la stessa struttura astratta: ad un record del file corrisponde un elemento (indicizzato) della tabella. Il numero di sequenza del record si trasforma in un valore dell'indice della tabella.
Con questo "caricamento" (in inglese "LOAD") del file, ci assicuriamo l'accesso diretto ad ogni elemento della tabella attraverso il meccanismo dell'indice.
Il caricamento avviene scandendo il file sequenziale e portanto in successivi elementi della tabella i record che mano a mano leggiamo. Per ogni record dobbiamo scomporre l'informazione in vari campi, e perciò dovremo predisporre tante variabili con indice quanti sono i campi del record. Ecco l'algoritmo:

```
INIZIO. Caricamento
  Apri file ELENCO
  RIPETI
  PER SEMPRE
    LEGGI record
    SE J>100 ALLORA ESCI
    Estrai campo 1, campo 2,
    campo 3 da record
    Metti campo 1, campo 2,
    campo 3 in tabella (J).
    J = J + 1
    SE EOF ALLORA ESCI
  FINE RIPETI
  Chiudi file
FINE.
```

L'algoritmo qui presentato permette di caricare dal file una tabella lunga 100 elementi. La sua realizzazione in BASIC è presentata nella parte terza del programma di figura 1. La frase "Estrai campo..." si traduce con l'istruzione BASIC MID$, che estrae una sottostringa di caratteri da una variabile alfanumerica.
La frase "Metti campo 1, campo 2, campo 3 in tabella (J)", si

traduce nelle tre istruzioni di BASIC 330-350, che impostano una serie di tre variabili con indice: FLD1$, FLD2$ e FLD3$, una per ogni campo.
L'algoritmo presentato carica un massimo di 100 elementi. Il limite della gestione di un file attraverso il caricamento in tabella consiste proprio in questo: non è sempre fisicamente possibile predisporre in memoria una tabella grande a sufficienza per contenere l'intero file. Per questa ragione esistono le strutture dei files ad accesso diretto e per chiave: se fosse sempre possibile (o conveniente) caricare l'intero file in memoria, la soluzione a tabella sarebbe quella più efficiente in termini di rapidità di reperimento dell'informazione.
Considerando files piccoli, come per esempio il nostro elenco telefonico, il problema dello spazio non pesa molto, e la soluzione di caricare l'intero file in tabella ci permette di recuperare l'informazione attraverso la scansione della tabella, possedendo una chiave di ricerca, per esempio il cognome dell'amico di cui vogliamo il numero di telefono.
L'algoritmo di recupero dell'informazione è il seguente:

```
INIZIO. Scansione tabella per
trovare elemento chiave.
 J = 1
 RIPETI
 FINCHÈ (chiave ≠ CAMPO2
 (J)) AND (J ≤ 100)
  J = J + 1
 FINE_RIPETI
 Stampa "Il numero di
 telefono di" chiave
 "corrisponde a" CAMPO3 (J)
FINE
```

Questo algoritmo, con alcuni controlli ulteriori che servono a impedire delle scelte fasulle, è implementato in BASIC in figura 2.

CONCLUSIONE

I files sono strutture dati esterne, attraverso le quali i dati vengono memorizzati su supporto magnetico (nastro o disco).
I tipi di accesso ad un file sono tre: sequenziale, diretto e diretto per chiave.
Solo il file sequenziale esiste come struttura dati esterna elementare: tutte le strutture file più complesse sono organizzate basandosi sulla memorizzazione di più di un file sequenziale.
Il file sequenziale, che è l'unico realizzabile su nastro, non permette il recupero di un record singolo: bisogna sempre scandire l'intero file.
Tuttavia, attraverso il CARICAMENTO dell'intero file in tabella (struttura dati interna), si possono realizzare accessi diretti o per chiave alle informazioni che interessano. Questa tecnica è però possibile solo per files piccoli, per ragioni di spazio.
Abbiamo infine visto un algoritmo di scansione di una tabella

per recuperare l'informazione (numero di telefono) sulla base di una chiave (cognome dell'amico); invitiamo il lettore a provare il programma BASIC per capirne bene il funzionamento, specialmente l'uso della variabile NEC, che porta il numero degli elementi caricati in tabella.

Marco

```
1 REM PARTE PRIMA: CREAZIONE FILE
10 OPEN 1,1,1,"ELENCO"
20 B$=" "
40 PRINT "■PER FINIRE SCRIVI ###" : PRINT : REM "■" = TASTO CLR HOME + SHIFT
50 INPUT "NOME ";NOME$
60 IF NOME$ = "###" GOTO 130
70 INPUT "COGNOME ";COG$
80 INPUT "NUMERO TELEFONICO ";NTEL$
90 NOME$ = NOME$+B$:NOME$ = LEFT$ (NOME$,10)
100 COG$ = COG$+B$+B$:COG$ = LEFT$ (COG$,25)
110 NTEL$ = NTEL$+B$:NTEL$ = LEFT$ (NTEL$,12)
160 REC$ = NOME$+COG$+NTEL$
170 PRINT#1,REC$
180 GOTO 40
185 PRINT "■RIAVVOLGI IL NASTRO E POI DIGITA 'CONT'"
190 CLOSE 1 : STOP
195 REM PARTE SECONDA: EMISSIONE FILE
200 OPEN2,1,0,"ELENCO"
210 PRINT "■CONTENUTO DEL FILE MEMORIZZATO:" : PRINT
220 INPUT#2, REC$: TS=ST: REM ST=STATUS
225 PRINT REC$
230 IF TS = 64 GOTO 270: REM 'STATUS 64' = EOF
250 GOTO 220
260 PRINT "■RIAVVOLGI IL NASTRO E POI DIGITA 'CONT'"
270 CLOSE 2 : STOP
280 REM PERTE TERZA: CARICAMENTO DEL FILE
285 DIM F1$(100),F2$(100),F3$(100)
290 OPEN3,1,0,"ELENCO"
295 J = 1
300 INPUT#3,REC$: TS = ST
320 IF J > 100 GOTO 390
330 F1$(J) = MID$ (REC$,1,10)
340 F3$(J) = MID$ (REC$,36,12)
350 F2$(J) = MID$ (REC$,11,25
360 IF TS = 64 GOTO 390
370 J = J+1
380 GOTO 300
390 CLOSE 3
400 FOR I = 1 TO J
410 PRINT F1$(I)
420 PRINT F2$(I)
430 PRINT F3$(I)
440 PRINT "- - - - - - - - - - - - - - - - - - - -"
500 NEXT I
```

```
280 REM PERTE TER2A: CARICAMENTO DEL FILE
285 DIM F1$(100),F2$(100),F3$(100)
290 OPEN3,1,0,"ELENCO"
295 J = 1
300 INPUT#3,REC$: TS = ST
310 IF J > 100 GOTO 390
320 FOR I = 1 TO NEC
330 F1$(I) = MID$ (REC$,1,10)
340 F3$(I) = MID$ (REC$,36,12)
350 F2$(I) = MID$ (REC$,11,25)
360 IF TS = 64 GOTO 390
370 J = J+1
380 GOTO 300
390 CLOSE 3
1000 GOSUB 2105
1100 INPUT "DIMMI IL COGNOME CHIAVE";COGN$
1200 CHIAVE$ = GOGN$ + "
1300 CHIAVE$ = LEFT$ (CHIAVE$,25)
1600 GOSUB 1900
1620 IF F = 1 THEN PRINT "IL NUMERO DI "COG$" E'" F3$(T)
1630 IF F = 0 THEN PRINT "IL COGNOME "COG$" NON FIGURA NELL'ELENCO"
1800 GOTO 1100
1900 REM  RICERCA DI UN ELEMENTO IN TABELLA
2000 F=0 : T=1 : J=1
2010 IF CHIAVE$ = F2$(J) GOTO 2070
2030 IF J > NEC THEN F = 0: GOTO 2080
2050 J = J+1
2060 GOTO 2010
2070 F = 1: T = J
2080 RETURN
2105 REM  EMISSIONE DELLA TABELLA
2110 FOR I = 1 TO NEC
2210 PRINT F1$(I)
2310 PRINT F2$(I)
2410 PRINT F3$(I)
2510 PRINT "- - - - - - - - - - - - - - - - - - - - -"
2610 NEXT I : RETURN
```

"COME TRADURRE UN NUMERO DECIMALE ALLA BASE 2 O 16"

I numeri in decimale ci sono familiari. Si dividono in 2 categorie: i numeri interi (senza virgola) e quelli con la virgola.
Ora vediamo come convertire un numero decimale (con o senza virgola!) in un numero di qualunque base compresa tra 2 e 16.
Il solito programma BASIC vi aiuterà ad approfondire i concetti esposti.

RAPPRESENTAZIONE DEI NUMERI

Nello scorso numero abbiamo parlato dei "Sistemi di Numerazione" decimale, binaria, ottale ed esadecimale.
Dopo esservi esercitati con il programma basic nelle diverse rappresentazioni, ora siete sicuramente in grado di destreggiarvi da un sistema di numerazione all'altro e magari avete anche imparato ad usare in modo più consapevole anche la familiare "rappresentazione decimale".
L'**informazione** all'interno dell'elaboratore è rappresentata in binario, all'esterno è invece rappresentata in decimale: il passaggio dell'informazione tra l'interno e l'esterno di un elaboratore è garantito dalle unità di ingresso ed uscita, che provvedono a convertire l'informazione da rappresentazione binaria a decimale e viceversa (di questi sistemi, noti come CODICI, ne parleremo in seguito).

Quando diventerete esperti programmatori vi capiterà d'aver bisogno di leggere alcune informazioni direttamente nella rappresentazione interna, per rimuovere errori od anche solo per curiosità.
Vediamo allora come è possibile passare dalla rappresentazione decimale alla rappresentazione binaria (il viceversa siete già in grado di farlo*).
Per convertire un numero intero decimale nel corrispondente numero binario (ottale od esadecimale) si utilizza il seguente metodo.

Metodo della divisione. Si divide il numero intero decimale per la "base" del sistema di numerazione a cui vogliamo passare (2, 8 o 16); si pone il resto della divisione come cifra binaria (ottale od esadecimale) meno significativa (da destra verso sinistra) del corrispondente numero binario che stiamo costruendo. Si divide poi il risultato della prima divisione ancora

per la "base" e si ripete il processo, ottenendo via via le altre cifre.
Il processo si arresta quando il risultato di una divisione diventa uguale a 0.
Vogliamo, per esempio, rappresentare il numero 183_{10} in binario, ottale ed esadecimale: di seguito riportiamo i "**risultati parziali**" delle divisioni ed il **resto**

BASE 2

VALORE INIZIALE E RISULTATI PARZIALI	RESTO
183 : 2	
91	1
45	1
22	1
11	0
5	1
2	1
1	0
0	1

$183_{10} = (10110111)_2$

BASE 8

	RESTO
183 : 8	
22	7
2	6
0	2

$183_{10} = (267)_8$

BASE 16

183 : 16	
11	7
0	B

$183_{10} = (B7)_{16}$

Abbiamo fin qua visto come trasformare un numero intero decimale in un numero binario (ottale ed esadecimale); ma come ben sapete i numeri interi non esauriscono l'insieme dei numeri che abitualmente usiamo: ci sono anche i numeri frazionari (es. 10.5). Di seguito analizzeremo come trasformare questi numeri; per quanto riguarda la parte intera del numero (es. 10.5, parte intera = 10, parte decimale = 5) viene trasformata secondo il metodo della divisione, illustrato nelle pagine precedenti; mentre la parte decimale viene convertita con il "**metodo della moltiplicazione**".

Questo metodo si può schematizzare come segue.
Si moltiplica la parte frazionaria del numero per 2 (8 o 16); si prende la parte intera del prodotto ottenuto e lo si pone come cifra binaria (ottale od esadecimale) più significativa (**attenzione** in questo metodo da sinistra verso destra) del corrispondente numero binario; si procede in modo iterativo moltiplicando la parte decimale del prodotto ottenuto, arrestandosi quando si raggiunge la precisazione desiderata.

Osservate che questo metodo non si arresta sempre a zero, come il precedente, ma siamo noi a dover decidere di arrestare la rappresentazione del numero alla terza, quarta...cifra decimale, cioè all'ultima **cifra** che riteniamo **significativa**.
Consideriamo il numero 0.250 di cui vogliamo la rappresentazione binaria; questo numero ha 3 cifre nella parte frazionа-

ria, decidiamo allora di arrestare il metodo della moltiplicazione anche alla terza cifra binaria (o prima se riscontriamo lo zero)

Ø.250	. 2	=	Ø.5	Ø
Ø.5			1.Ø	1
Ø.Ø			Ø	Ø

$(Ø.25Ø)_{10} = (Ø.Ø1Ø)_2$

Anche in questo numero il programma basic vi aiuterà a capire meglio quanto spiegato nelle pagine precedenti, permettendovi di verificare in ogni momento se il numero binario che avete ottenuto è realmente la conversione del numero decimale di partenza; esercitatevi e sarete allora pronti per affrontare il prossimo argomento. Aritmetica binaria.

IL PROGRAMMA BASIC

Vediamo la realizzazione in BASIC della conversione da DECIMALE ad altra base per numeri interi.
Come limite non convertiamo numeri che nella nuova base abbiamo più di 8 cifre.
Il programma è la realizzazione BASIC del metodo illustrato nelle pagine precedenti. Tale metodo si basa, come abbiamo visto nelle pagine precedenti, su successive divisioni e determinazioni di altrettanti resti. La sequenza dei resti messi uno accanto all'altro (nell'ordine inverso a cui sono generati), è la rappresentazione del numero nella nuova base. Perciò ci si serve di una variabile R(I) per contenere le cifre del numero nella nuova base.
La prima volta il numero da dividere (dividendo) è DEC, cioè il numero introdotto (istruzione 3Ø).
Le volte successive il quoziente della divisione precedente diventa il nuovo dividendo. Ricordiamo che il quoziente è l'intero più grande contenuto nel risultato della divisione, e che il divisore è la base prescelta (istruzione 2Ø).
In sostanza il quoziente si trova con l'istruzione 6Ø e viene memorizzato in A. (Osservate che la prima volta Q è uguale a DEC). Il resto della divisione è poi calcolato dall'istruzione 7Ø.
L'istruzione 75 imposta il nuovo dividendo (Q) al valore del quoziente della divisione (A).
La sequenza di istruzioni 6Ø-75 vengono ripetute (in virtù di 6Ø, 8Ø e 9Ø) finché Q non diventa O.
Il risultato viene presentato emettendo le cifre trovate scorrendo R(I) all'indietro.
Ultimo commento: poiché R(I) contiene numeri e non cifre di sistemi in base superiore a 1Ø, è necessario per queste cifre, emettere invece i caratteri A,B, C etcetera come già spiegato nell'articolo precedente.
Questo ci induce a considerare ogni cifra come rappresentata da un carattere e costruire il vettore di conversione S$(I). Osservate che poiché R(I) può valore O, e O non può essere indice (istruzione 11Ø), bisogna ag-

giungere 1 a R(I) per ottenere la cifra corrispondente nel vettore di conversione.

MGD

```
10 DIMR(8),S$(16)
13 S$(1)="0":S$(2)="1":S$(3)="2":S$(4)="3":S$(5)
   ="4":S$(6)="5"
15 S$(7)="6":S$(8)="7":S$(9)="8"
16 S$(10)="9":S$(11)="A":S$(12)="B":S$(13)="C"
   :S$(14)="D":S$(15)="E":S$(16)="F"
20 INPUT" INTRODUCI BASE ";BASE
30 INPUT" INTRODUCI NUMERO INTERO :";DEC
40 Q=DEC:I=1:R(1)=0
50 IFQ=0GOTO99
60 A=INT(Q/BASE)
70 R(I)=(Q-(A*BASE))
75 Q=A
80 I=I+1
90 GOTO50
99 REM FINE RIPETI
100 FORK=ITO1STEP-1
110 PRINTS$(R(K)+1);
120 NEXTK

READY.
```

SET ISTRUZIONI DI CARICAMENTO

Ecco finalmente giunto il momento di considerare alcune istruzioni del L/M. In questa prima parte esamineremo le istruzioni di caricamento e trasferimento dati, che svolgono il ruolo più importante in un programma.

3.1 ISTRUZIONI DI CARICAMENTO

In un programma (in qualsiasi linguaggio esso sia interpretato) è importantissima la capacità di potere scambiare e manipolare dati. Nel l/m non abbiamo variabili come in linguaggi di alto livello (quale il BASIC), ma gli accumulatori, i registri e le locazioni di memoria le sostituiscono altrettanto bene. È perciò molto importante uno scambio reciproco di dati tra memoria e registri. Nelle istruzioni di caricamento esiste sempre un Destination Address, il "luogo" in cui un dato deve essere trasferito. Nel codice mnemonico, per le istruzioni di caricamento si usano due prefissi: "LD" ed il registro da usare nel trasferimento; da memoria a registro; "ST" ed il registro coinvolto in un trasferimento da registro a memoria.
Facciamo un esempio: supposto nnnn uguale ad un indirizzo di 16 bits, per trasferire un dato da una locazione rappresentata da tale indirizzo all'Accumulatore avremo:

```
LDA nnnn                    nnnn = indirizzo 16 bits
```

Il Destination Address del trasferimento è l'Accumulatore, il Source Addr. è la locazione nnnn, ed il trasferimento è registro ←memoria. Naturalmente se avessimo voluto indirizzare il contenuto di nnnn verso i registri X o Y avremmo scritto:

```
STX nnnn        STY nnnn        nnnn = indirizzo 16 bits
```

Per avere un trasferimento di tipo registro →memoria, cioè trasferire un dato da un registro ad una locazione di memoria, al posto di "LD" useremo "ST", per cui se volessimo mettere il contenuto dell'Accumulatore nella locazione nnnn avremo:

```
STA nnnn                    nnnn = indirizzo 16 bits
```

ed analogamente per X ed Y:

```
STX nnnn        STY nnnn        nnnn = indirizzo 16 bits
```

Tratteremo singolarmente i vari metodi di caricamento riferiti a ciascun registro, in quanto non tutti i modi di indirizzamento sono possibili per ogni registro. Inoltre useremo la seguente notazione:

```
nnnn  = indirizzo assoluto a 16 bits
nn    = indirizzo ad 8 bits
dd    = dato di 8 bits
```

3.2.1 CARICAMENTO TRAMITE ACCUMULATORE

SUFFISSI: LDA e STA

Le istruzioni che si riferiscono all'Accumulatore posseggono la maggior parte dei modi di indirizzamento possibili.

3.2.1.1 IMMEDIATE ADDRESSING

```
LDA #dd ;questo tipo di indirizzamento
        ;carica direttamente il numero dd
        ;in A. È possibile solo il LOAD
```

3.2.1.2 ABSOLUTE ADDRESSING

```
LDA nnnn ;il trasferimento avviene tra
         ;la locazione nnnn e
STA nnnn ;l'Accumulatore
```

3.2.1.3 ZERO PAGE ADDRESSING

```
LDA nn   ;l'indirizzo della locazione coinvolta
         ;nel trasferimento è in pagina zero.
STA nn   ;nn rappresenta il byte basso di un
         ;indirizzo assoluto compreso tra 0000 e 00nn
```

3.2.1.4 INDEXED INDIRECT ADDRESSING

```
LDA (nn,X) ;il contenuto del registro X è sommato
           ;al byte nn. Il risultato della somma punterà
STA (nn,X) ;ad una locazione in pagina zero, contenente il
           ;byte basso (la successiva quello alto), di un
           ;indirizzo a 16 bits da usare nel trasferimento
```

3.2.1.5 INDIRECT INDEXED ADDRESSING

```
LDA (nn),Y ;il byte nn punta ad una locazione in pagina zero.
           ;tale locazione e la sua successiva contengono un
```

```
STA (nn),Y ;indirizzo a 16 bits, cui verrà sommato il contenuto
           ;del registro Y, per ottenere l'indirizzo effettivo
```

3.2.1.6 INDEXED ZERO PAGE A ADDRESSING [X]

```
LDA nn,X  ;l'indirizzo effettivo è rappresentato dalla
          ;somma del byte nn e del contenuto di X. Questo
STA nn,X  ;indirizzamento punta sempre in pagina zero ed
          ;è possibile solo usando il registro X.
```

3.2.1.7 INDEXED ABSOLUTE ADDRESSING

```
LDA nnnn,X ;uguale al punto 3.2.1.6, ma con indirizzo a 16 bits e non
STA nnnn,X ;più in pagina zero. Inoltre possibile sia con X che con Y
           ;
LDA nnnn,Y ;
STA nnnn,Y ;
```

3.2.2 CARICAMENTO TRAMITE X

A questo punto citeremo solo il tipo di indirizzamento possibile, il suo codice mnemonico e, se non vi saranno commenti, ci si potrà riferire a quelli contenuti nei punti da 3.2.1.1 a 3.2.1.7.

3.2.2.1 IMMEDIATE ADDRESSING

```
LDX #dd    ;
```

3.2.2.2 ABSOLUTE ADDRESSING

```
LDX nnnn   ;
           ;
STX nnnn   ;
```

3.2.2.3 ZERO PAGE ADDRESSING

```
LDX nn     ;
           ;
STX nn     ;
```

3.2.2.4 INDEXED ABSOLUTE ADDRESSING [Y]

```
LDX nnnn,Y ;come al punto 3.2.1.7, ma essendo X coinvolto
           ;nel trasferimento, è possibile usare solo
STX nnnn,Y ;il registro Y come indice.
```

3.2.2.5 INDEXED ZERO PAGE ADDRESSING [Y]

```
LDX nn,Y    ;come al punto 3.2.2.4, ma indirizzamento
            ;in pagina zero.
STX nn,Y    ;
```

3.2.3 CARICAMENTO TRAMITE Y

3.2.3.1 IMMEDIATE ADDRESSING

```
LDY #nn     ;
```

3.2.3.2 ABSOLUTE ADDRESSING

```
LDY nnnn    ;
            ;
STY nnnn    ;
```

3.2.3.3 ZERO PAGE ADDRESSING

```
LDY nn      ;
            ;
STY nn      ;
```

3.2.3.4 INDEXED ABSOLUTE ADDRESSING

```
LDY nnnn,X  ;come al punto 3.2.2.4, ma con ruolo dei
            ;registri invertito
STY nnnn,X  ;
```

3.2.3.5 INDEXED ZERO PAGE ADDRESSING [X]

```
LDY nn,X    ;
            ;
STY nn,X    ;
```

3.3 ISTRUZIONI DI SCAMBIO

Abbiamo fino ad ora esaminato istruzioni di scambio dati tra me-

moria e registri. Esistono però anche istruzioni che permettono lo scambio di dati tra registri, specificate dalla lettera "T", seguita dai due registri coinvolti nello scambio di informazioni.
Per esempio, se vogliamo che il contenuto di A passi in X scriveremo:

```
TAX
```

Vediamo nell'esempio che alla "T" segue il registro sorgente, da cui prelevare il dato, ed in seguito il registro destinazione.

Le possibili istruzioni di scambio sono:

```
TAX    ;trasferimento ad A ad X (A→X)
TAY    ;trasferimento da A ad Y (A→Y)
TSX    ;trasferimento da S (Stack Pointer) ad X (S→X)
TXA    ;trasferimento da X ad A (X→A)
TXS    ;trasferimento da X ad S (X→S)
TYA    ;trasferimento da Y ad A (Y→A)
```

3.4 COINVOLGIMENTO DEI FLAGS

Anche se ciò potrà risultare incomprensibile a chi inizia ad avventurarsi nel l/m, per dovere di informazione verso chi magari è già al corrente del funzionamento dei flags (non ne abbiamo ancora parlato), diciamo che nelle istruzioni di trasferimento registro ↔ memoria o registro ⟷ registro sono coinvolti i flags di zero e di segno. Unica eccezione in cui non è coinvolto alcun flag è l'istruzione "TXS".

Giunti a questo punto, prima di chiudere la puntata, pensiamo sia opportuno fornirvi alcuni esempi di paragone tra BASIC e l/m, per rendervi più chiaro il funzionamento delle istruzioni di quest'ultimo.
Se per esempio volete trasferire dei bytes da una zona di memoria all'altra, in BASIC potreste avere:

```
10 A = PEEK 1671
20 POKE 1897,A
30 A = PEEK 1672
40 POKE 1898,A
```

Avete così trasferito due bytes dalle locazioni 1671 e 1672 alle locazioni 1897 e 1898. L'equivalente in l/m potrebbe essere:

```
1 LDA 1671 ;riempi A col contenuto della locazione 1671
2 STA 1897 ;metti nella locazione 1897 il contenuto di A
3 LDX 1672 ;riempi X col contenuto della locazione 1672
4 STX 1898 ;metti nella locazione 1898 il contenuto di X
```

Il fatto che per i punti 1 e 2 abbiamo usato l'Accumulatore ed ai punti 3 e 4 il registro X, è solo un esempio di come sia possibile usare qualsiasi registro in questo caso. Si sarebbe potuto benissimo avere:

```
1 LDA 1671 ;
2 STA 1897 ;
3 LDA 1672 ;
4 STA 1898 ;
```

senza che nulla cambiasse.
Se invece in BASIC avessimo avuto:

```
10 X=0
20 A=PEEK 1671+X
30 POKE 1897+X,A
40 X=1
50 A=PEEK 1671+X
60 POKE 1897+X,A
```

avremmo ottenuto sempre lo stesso trasferimento.
In l/m avremmo potuto scrivere:

```
1 LDX #0        ;carico X con 0
2 LDA 1671+X    ;riempio A col contenuto della locazione 1671+
                 X=1671
3 STA 1897+X    ;metto A nella locazione 1897+X=1897
4 LDX #1        ;riempio X con 1
5 LDA 1671,X    ;riempio A con la locazione 1671+X=1672
6 STA 1897,X    ;carico 1897+X=1898 col contenuto di A
```

In questo caso abbiamo usato un indirizzo assoluto indicizzato, che però risulta più complesso rispetto al primo esempio. Teniamo a precisare che il fatto che le variabili BASIC si chiamino X ed A è solo una semplificazione perché avremmo anche potuto chiamarle UNO e DUE. Ciò non è possibile però in l/m, in quanto X,Y ed A sono i nomi dei registri e non delle variabili.

A.V.S.

GLI ALGORITMI FONDAMENTALI

In questo numero tratteremo alcune tecniche fondamentali di programmazione, tecniche che ogni buon programmatore dovrebbe conoscere.
Prima di iniziare una trattazione più particolareggiata è bene ribadire alcuni concetti fondamentali.
Innanzitutto precisiamo che un "ALGORITMO" non è altro che un processo di elaborazione chiaramente definito e implementabile su qualunque macchina. Per facilitarne la comprensione noi elaboreremo le procedure in linguaggio BASIC, ma una volta capiti i principi di funzionamento è possibile svilupparle in qualunque linguaggio.
Il primo problema che affrontiamo riguarda l'ordinamento di un insieme di elementi che noi consideriamo come un vettore, numerico o alfanumerico. In molti casi infatti un programmatore si troverà alle prese con una schiera di nomi o numeri disposti più o meno alla rinfusa che devono necessariamente essere ordinati per permettere una ricerca più rapida o per altri motivi di ordine pratico.
A questo scopo sono stati sviluppati numerosi algoritmi di ordinamento, più o meno veloci, che si adattano alle diverse esigenze; il più conosciuto e usato è comunque il famoso "BUBBLE SORT", cioè l'ordinamento a bolla.
Ma cerchiamo ora di capire come funziona questo semplice ma efficace algoritmo.
Innanzitutto bisogna sapere da quanti elementi è composto il vettore, quindi dovremo provvedere a creare un ciclo che vada da 1, il primo elemento, fino alla lunghezza del vettore stesso meno 1. All'interno di questo ciclo dobbiamo crearne un altro simile e, all'interno di quest'ultimo, dovremo confrontare l'elemento puntato dalla variabile di controllo con quello immediatamente successivo; se l'elemento successivo è minore di quello precedente allora dovremo trasferire il contenuto dell'elemento puntato in una variabile di comodo che ci servirà esclusivamente per salvare il contenuto di quest'ultimo, dopodiché porremo nell'elemento puntato il valore dell'elemento successivo e quindi in quest'ultimo metteremo il contenuto della variabile di comodo che conteneva appunto il valore dell'elemento puntato. A questo punto si conclude il ciclo interno.
Vediamo comunque il risultato pratico delle nostre dissertazioni.

```
 10 FOR A = I TO L-1
 20 Q = 0
 30 FOR B = 1 TO L-1
 40 IF V(B + 1) > = V(K)
    THEN GOTO 90
 50 C = V(B)
 60 V(B) = V(B + 1)
 70 V(B + 1) = C
 80 Q = 1
 90 NEXT B
100 IF Q = 0 THEN GOTO 120
110 NEXT A
120 STOP
```

Come avrete capito la variabile L contiene il numero di elementi interessati all'ordinamento, mentre C è la cosiddetta variabile di comodo.
A questo punto vi chiederete a cosa può servire la variabile Q. Ebbene, noterete che dopo ogni "spazzolata" del ciclo interno questa variabile viene posta uguale a zero e assume il valore 1 solo qualora si verifichi qualche scambio di elementi nel ciclo interno. È quindi evidente che se non avviene nessuno scambio essa mantiene il suo valore originale, ma in questo caso significa che gli elementi sono già in ordine e quindi è inutile continuare le operazioni di controllo; a questo scopo, alla fine del ciclo interno vi è una verifica su questa variabile e, nel caso essa risulti uguale a zero, avviene un salto alla linea 120 che termina l'esecuzione del programma.
Se non siete ancora convinti del modo in cui opera il programma facciamo un breve esempio pratico limitato ovviamente a soli 3 elementi. Poniamo che inizialmente gli elementi siano disposti come segue:
2, 9, 7
Il primo confronto controlla se il secondo elemento è maggiore del primo e poiché ciò è vero salta alla linea 90 che conclude il ciclo; il secondo passaggio confronta il terzo elemento con il secondo e poiché quest'ultimo è maggiore del terzo viene effettuato lo scambio fra i due elementi, cosicché il terzo prende il posto del secondo e viceversa. A questo punto termina la prima "spazzolata" interna che ha avuto anche l'effetto di alterare la variabile di controllo Q che ora è uguale a 1.
Ora gli elementi sono così disposti:
2, 7, 9
Come potete notare essi sono già in ordine ma il programma non lo sa ancora. Occorrerà attendere il prossimo passaggio, ove non avverranno scambi e quindi la variabile Q resterà a zero; il successivo confronto alla linea 100 terminerà quindi l'esecuzione.
Bene, nonostante il semplicissimo esempio, dovreste esservi convinti dell'efficacia dell'algoritmo e del semplice principio di funzionamento. Naturalmente questo non è l'unico sistema di ordinamento, ma come abbiamo già detto è sicuramente il più famoso e il più usato. Ricordiamo infine che noi abbiamo effettuato confronti tra variabili numeriche, ma in molti casi è possibile eseguire anche

confronti tra elementi alfanumerici.
Ora però qualcuno di voi potrebbe chiedersi se vale realmente la pena di sprecare memoria e tempo di esecuzione solo per il piacere di ottenere un vettore ordinato.
In primo luogo ribadiamo che questo è essenzialmente un utile esercizio di programmazione, comunque a prescindere da ciò, l'utilità di avere un vettore ordinato si farebbe sentire qualora avessimo la necessità di ricercare nel minor tempo possibile un elemento della lista.
Il sistema più semplice e intuitivo per cercare un elemento in una lista è procedere dall'inizio esaminando un elemento per volta. Questo tipo di ricerca si chiama "ricerca sequenziale" e ovviamente ha una scarsa efficacia in quanto se dobbiamo cercare l'ultimo elemento della lista dobbiamo necessariamente passare tutti quelli che lo precedono; questo potrebbe anche essere accettabile nel caso si tratti di pochi elementi, ma se prendiamo in considerazione liste di centinaia o addirittura migliaia di elementi è evidente che si hanno dei tempi di ricerca assolutamente inaccettabili. Se invece disponiamo di una lista ordinata, come quella vista in precedenza, possiamo applicare un altro algoritmo di ricerca chiamato "ricerca binaria" il quale riduce in maniera drastica (o più precisamente logaritmica), il numero medio di confronti necessari per rintracciare l'elemento desiderato. Infatti usando la tecnica di ricerca sequenziale il numero medio di confronti corrisponde molto approssimativamente alla metà del numero totale di elementi, mentre usando la ricerca binaria il numero massimo di confronti richiesti è uguale al logaritmo in base 2 del numero totale di elementi più uno. Ad esempio su una lista di 500 elementi il LOG_2 di 500 è 8, quindi il numero massimo di confronti sarà 9 contro i 250 medi richiesti dal sistema sequenziale.
Ma ora vediamo in pratica come è strutturato l'algoritmo di ricerca binaria che può essere applicato solo a liste ordinate.
Innanzitutto usando questo sistema, la ricerca non partirà dall'inizio, ma dalla metà della lista: a questo punto se l'elemento da cercare è minore, si scarta già la metà inferiore della lista, o viceversa, se è superiore, si scarta la parte superiore. Ora la lista ha già dimezzato le sue dimensioni originali, quindi si prende in esame l'elemento centrale della parte rimanente e si scarta ancora metà della lista rimanente; il tutto prosegue finché si trova l'elemento desiderato. Adesso capirete certamente perché la lista deve essere ordinata, altrimenti non si potrebbero scartare a priori degli elementi senza averli esaminati.
Vediamo anche in questo caso

un semplice esempio di ricerca binaria effettuata su una tabella ordinata di 10 numeri che vanno appunto dall'uno al dieci, volendo trovare il numero 4.

In questo caso vengono effettuati ben quattro confronti e quindi il vantaggio sulla ricerca sequenziale non è apprezzabile ma lo diviene nel caso di liste composte da un migliaio di elementi, dove sono sufficienti 10 confronti al massimo contro i 500 richiesti mediamente dalla ricerca sequenziale.
Ma vediamo ora una implementazione in BASIC di questo vantaggioso algoritmo di ricerca.

```
10 S = 1:D = N
20 P = INT((S + D)/Z)
30 IF D < S THEN GOTO 70
40 IF A$ = B$(P) THEN PRINT
   "TROVATO":STOP
50 IF A$ > B$(P) THEN D = P-1:
   GOTO 20
60 5 = P + 1:GOTO 20
70 STOP
```

Nonostante l'apparente complessità, questo breve programma non è altro che una applicazione pratica della ricerca binaria. Come potete vedere sono necessarie 3 variabili per effettuare i calcoli necessari; la variabile S punta al primo elemento, mentre la variabile D punta all'ultimo, in quanto contiene il numero N di elementi presenti nella lista. La variabile P è quella usata per puntare l'elemento da confrontare del vettore B$(), mentre A$ contiene il nome da trovare. Inizialmente la variabile P punterà proprio all'elemento centrale: infatti ponendo di avere una lista di N = 100 elementi, P sarà uguale alla parte intera di (1 + 100)/2 che è uguale a 50. Possiamo quindi notare che se l'elemento puntato è maggiore dell'elemento da trovare, la riga 50 provvede a escludere dalla ricerca tutta la lista a partire da quell'elemento; analogamente avviene se l'elemento puntato è minore.
La linea 30 controlla se D < S, eventualità che può verificarsi solo se l'elemento da cercare non esiste nella lista; in questo caso il programma salta alla linea 70 che termina l'esecuzione. La linea 40 invece controlla se l'elemento puntato e quello dato coincidono, nel qual caso stampa un semplice messaggio di conferma.
Bene, terminiamo qui le nostre dissertazioni sugli algoritmi di ordinamento e ricerca, nella speranza che vi siano di aiuto per migliorare sempre più la vostra abilità nel campo della programmazione.

Massimo Cellini

ISTRUZIONI PER LA CASSETTA DI PLAY ON TAPE COMPUTER N. 8

ISTRUZIONI DI CARICAMENTO

Prendete la cassetta, mettetela nel registratore con la facciata in alto. Per i possessori di VIC 20. Accendere poi il computer e premere "SHIFT" e "RUN STOP" contemporaneamente; per i possessori di C 16 e di Plus 4 occorre invece digitare LOAD e premere sul tasto RETURN: verrà in questo modo caricato il primo programma residente su nastro: 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'. 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)

METEORE

Mentre navighi nello spazio siderale centinaia di meteoriti e di razzi nemici ti corrono incontro cercando di distruggerti; dovrai districarti abilmente in quel labirinto di meteore cercando di non rimanere colpito. Riuscirai a dimostrare la tua abilità ottenendo un punteggio molto alto che alla fine del gioco comparirà sullo schermo.

Inserendo la tua cassetta appariranno le istruzioni:

F7 per attivare la potentissima arma a raggio laser che hai a disposizione;
A muove la tua astronave verso sinistra;
D muove la tua astronave verso destra.

Ora sei in possesso di tutte le indicazioni necessarie per iniziare la tua battaglia spaziale e a noi non resta che augurarti buon divertimento!

THE LAST BATTLE

Preparati a combattere una battaglia all'ultimo sangue contro pericolosissimi nemici provenienti da altre galassie; questi cercheranno di colpirti per distruggere lo schermo energetico che protegge la tua astronave: difenditi finché sei ancora in possesso di tutte le tue forze!

Per muoverti e per sparare usa i tasti:

F5 per sparare in alto
F7 per sparare in basso
Z per spostarti verso sinistra
X per spostarti verso destra.

Sul video apparirà l'immagine del campo di battaglia in cui dovrai combattere protetto dalla barriera energetica. I nemici si lanceranno con le loro astronavi contro le tue difese, cercando di distruggere la barriera: cerca di colpirli il più velocemente possibile.

Sullo schermo in alto potrai controllare in ogni momento il punteggio effettuato e l'energia rimasta a tua disposizione durante l'attacco.

Quando la battaglia avrà termine, comparirà il punteggio totale ottenuto e il record da raggiungere.

Per continuare a giocare rispondi S alla domanda del computer. Buon divertimento!

SEVEN ELEVEN

È il gioco d'azzardo che più appassiona i frequentatori di tutti i grandi Casinò. Questa volta potrai sfidare il tuo computer in una divertente partita a due.

Il capitale iniziale tuo e del computer ammonta a 100 $ a testa: ad ogni tiro vengono messi in gioco 10 $ che verranno aggiunti o detratti a seconda delle vincite o perdite.

Chi tiene il banco dovrà tirare per primo i dadi: con le combinazioni ottenute, il 12, il 2 e il 3 perdono immediatamente; il 7 e l'11 vincono. Se uscirà uno dei numeri rimasti, il giocatore dovrà rilanciare i dadi fino ad ottenere per la seconda volta il numero precedentemente uscito: se però, durante i successivi lanci, esce il 7, perde.

In caso di perdita il banco passa in mano all'avversario: se vinci invece potrai scegliere se continuare a tenere il banco (premi allora il tasto S alla domanda del computer) o passarlo (tasto N).

Per iniziare a giocare premi un tasto: in bocca al lupo e buon divertimento!

TELEFONO

È proprio vero che tra le spese da affrontare ogni mese, quella del telefono spesso non è la più indifferente, anche perché difficilmente programmabile e controllabile. Così si rischia di ritrovarsi, alla fine, con bollette telefoniche inaspettatamente esagerate.

Abbiamo pensato di offrirvi un comodo programma che possa aiutarvi giornalmente in un rapido calcolo di spesa ad ogni telefonata, per consentirvi una migliore pianificazione delle vostre uscite.

Caricato il programma "TELEFONO", sullo schermo del vostro VIC 20 apparirà la richiesta dell'ora e della distanza, espressa in chilometri, per definire le telefonate urbane o interurbane.

Dovrete successivamente specificare, per un calcolo più preciso, se la telefonata viene fatta in un giorno feriale, festivo o il Sabato, tenendo conto che le tariffe, nei giorni particolari, sono differenziate.

Componete quindi il numero telefonico che vi interessa e, alla risposta, premete un tasto. Considerando che uno scatto del contatore avviene in un periodo di tempo differente a seconda della distanza, del giorno, ecc., il computer vi segnalerà il tempo che trascorre e il numero di scatti effettuati.

Alla fine della telefonata premete nuovamente un tasto qua-

lunque e otterrete il tempo complessivo (in secondi) e il totale degli scatti.

Premendo ancora un tasto, apparirà la tabella dei costi in base al numero di scatti bimestrali effettuati a seconda se possedete un telefono singolo o duplex.

RIFLESSOMETRO

Vuoi sapere con sicurezza se possiedi riflessi pronti?

Allora prova immediatamente questo nuovo programma del tuo VIC 20. Per 5 volte successive comparirà sul video la scritta "ora!": tu dovrai premere il più rapidamente possibile il tasto F1. Apparirà un animale che definisce il grado di velocità raggiunto: lepre, topo, tartaruga, lumaca d'oro, lumaca d'argento e lumaca di bronzo. Al termine delle cinque prove il computer ti darà il risultato totale, definito dalla media dei singoli risultati. Stai bene attento a non premere F1 in anticipo, altrimenti il tuo VIC 20 ti inviterà a mantenere la calma per continuare la prova.

Devi essere rapido e controllato e non scoraggiarti ai primi tentativi: allenati costantemente e vedrai che in breve tempo anche tu diventerai una velocissima "lepre"!

(LATO B: C.16 - PLUS/4)

RIFLESSOMETRO

Mediante questo simpaticissimo programma potrai misurare la prontezza dei tuoi riflessi ed eventualmente impegnarti in una divertente gara di velocità con i tuoi amici.

Dopo aver caricato il programma, inserisci i nomi dei partecipanti al gioco: a questo punto sei pronto per iniziare. Sullo schermo apparirà un semaforo con luce rossa; quando la luce diventerà verde, dovrai essere pronto a schiacciare rapidamente un tasto qualunque: vedrai allora comparire il tempo impiegato espresso in sessantesimi di secondo.

Hai a disposizione cinque prove per misurare la prontezza dei tuoi riflessi: sul video apparirà anche la media dei tempi effettuati nelle varie prove.

Sei animali indicheranno rispettivamente il risultato di ogni prova; a secondo della velocità potrai essere paragonato ad una lepre, ad un topo, ad una tartaruga oppure ad una lumaca di bronzo, una lumaca d'argento o una lumaca d'oro. Oltre un certo tempo limite non esiste un animale per commentare la tua lentezza!

Attenzione a non schiacciare il tasto prima che il semaforo sia verde altrimenti il computer ti ammonirà con la scritta "Troppa fretta".

Al termine delle cinque prove otterrai il risultato complessivo che potrai paragonare con quello dei tuoi amici, e il tempo record ottenuto nella giornata.

Per ricominciare a giocare, premi S quando sul video appare la scritta: "Ancora".

Sei pronto per iniziare? Allora, via!

QUADRATI

È un gioco divertente che premia la furbizia e l'attenzione: consiste in una partita tra due giocatori il cui scopo è costruire il maggior numero possibile di quadrati, impedendo nello stesso tempo all'avversario di disegnare i suoi.

Per prima cosa, inserite i vostri nomi: ad entrambi i giocatori verrà assegnato il proprio colore, bianco o nero, con cui individuare i quadrati costruiti da ognuno. Per disegnare un quadrato, dovrete dare volta per volta al computer le coordinate x e y del lato che vi interessa. Il giocatore che riesce a costruire un quadrato completo, ha diritto alla mossa successiva; se anche questa volta riuscirà a disegnare un altro quadrato, potrà giocare ancora una mano prima che il gioco passi all'avversario.

Dopo le prime 60 mosse (30 per ogni giocatore) avrete la possibilità di tracciare i segmenti negli spazi già occupati dal vostro avversario, fino ad appropriarvi anche dei suoi quadrati.

Sullo schermo del computer appare continuamente il punteggio effettuato dai singoli giocatori. La partita termina dopo 100 mosse (50 per ogni giocatore) e il C16 vi darà subito il nome del vincitore.

Se volete sfidare nuovamente il vostro avversario, premete il tasto S, altrimenti N. Buon divertimento.

C16 MEMORY

È un utile passatempo, adatto per allenare la memoria, da provare da solo o con gli amici in una divertente gara. È consentito un massimo di 10 giocatori. Dopo aver caricato il programma, premete un tasto per iniziare: il computer vi chiederà il numero dei partecipanti e il loro nome. Inserite i dati, premendo subito dopo il RETURN.

Apparirà una griglia formata da 10 righe (numerate da 0 a 9) e 9 colonne (da 0 a 8). Le caselle coperte nascondono diversi simboli che dovrete cercare di scoprire formando coppie uguali.

Per scoprire una casella dovrete fornire le coordinate x = riga e y = colonna. Quando un giocatore riesce a formare una coppia, continua a giocare finché non commette un errore e il gioco passa in mano al giocatore successivo.

Quando tutte le caselle saranno state scoperte in coppia, il computer fornirà il nome del vincitore della gara e il numero delle coppie effettuate dai singoli partecipanti.

Buon divertimento!

CONTATORE TELEFONICO

È proprio vero che tra le spese da affrontare ogni mese, quella del telefono spesso non è la più indifferente, anche perché difficilmente programmabile e controllabile. Così si rischia di ritrovarsi, alla fine, con bollette telefoniche inaspettatamente esagerate. Abbiamo pensato di offrirvi un comodo programma che possa aiutarvi giornalmente in un rapido calcolo di spesa ad ogni telefonata, per consentirvi una migliore pianificazione delle vostre uscite.

Caricato CONTATORE TELEFONICO, sullo schermo del vostro C16 apparirà la richiesta dell'ora e della distanza, espressa in chilometri, per definire le telefonate urbane o interurbane. Dovrete inoltre specificare, per un preciso calcolo, se la telefonata viene fatta in un giorno feriale, festivo o il Sabato, tenendo conto che le tariffe, in questi periodi di tempo, sono differenziate.

Componete quindi il numero telefonico che vi interessa e, alla risposta, premete un tasto: il computer vi segnalerà il tempo che trascorre e il numero degli scatti effettuati. Alla fine della telefonata premete nuovamente un tasto qualunque e otterrete il tempo complessivo (in secondi) e il totale degli scatti.

Premendo ancora un tasto, apparirà la tabella dei costi in base al numero di scatti bimestrali effettuati a seconda se possedete un telefono singolo o duplex.

DAMA

Con questo programma potrete misurare la vostra abilità nel gioco della dama italiana sfidando il vostro computer, un avversario davvero preparato e che soprattutto non perdona il minimo errore!

La regole sono quelle della dama classica, ma ricordatevi che nella dama italiana non avete l'obbligo di mangiare le pedine dell'avversario: in questo modo, naturalmente, neanche le vostre pedine verranno "soffiate".

Si vince per blocco o per punteggio: ogni pedina vale un punto e ogni dama 3 punti.

Per muovere le pedine dovrai indicare, alla domanda del computer, la casella di partenza (lettera + numero); premi RETURN e poi scrivi anche la casella in cui desideri che la tua pedina venga spostata. Sul video appare a sinistra l'immagine della dama: a destra avrai in alto il conteggio delle mosse effettuate e in basso il numero delle pedine ancora a tua disposizione (lettera T) o a disposizione del computer (lettera C).

Le tue pedine sono quelle rappresentate da un tondino pieno; quelle avversarie sono forate al centro.

Ora siete in grado di iniziare la partita col vostro temibile avversario. Concentrazione quindi e...buon divertimento!

Numeri già apparsi:

VIDEOTECA COMPUTER N. 1
per i possessori di Commodore 64

Nel manuale n. 1: I tasti funzionali del Commodore 64 - Pseudocodice e programmazione Basic - Il joystick.

Nella cassetta n. 1: Slalom - Slot Machine - Bilancio familiare - Briscola - Domino.

VIDEOTECA COMPUTER N. 2
per i possessori di Commodore 64

Nel manuale n. 2: Il basic più veloce - Una migliore gestione del video per il 64 - Disegnare con tastiera e joystick - Pseudocodice: 2ª lezione.

Nella cassetta n. 2: Tennis 3d - Totocalcio - Gestione magazzino - Wargame - Colour search.

VIDEOTECA COMPUTER N. 3
per i possessori di Commodore 64

Nel manuale n. 3: I cicli annidati del Basic - Come sviluppare un programma (Squash e Trampolino) - Conosci il tuo CBM 64?

Nella cassetta n. 3: Starway - Easyword - Poker - Forza 4 - Test Quoziente Intellettuale.

VIDEOTECA COMPUTER N. 4
per i possessori di Commodore 64

Nel manuale n. 4: Pseudocodice: Istruzioni read e data - Figura richiama - Grafico a barre. Il basic del CBM 64. I linguaggi di programmazione. Come personalizzare i programmi.

Nella cassetta n. 4: Dieta - Calorie dei cibi - Visischool - Sink - Hat in the ring.

VIDEOTECA COMPUTER N. 5
per i possessori di Commodore 64

Nel manuale n. 5: Pseudocodice: 5° lezione - I sistemi di numerazione - Le periferiche di ingresso e uscita. Basic: I cicli.

Nella cassetta n. 5: Esherland - Bosco incantato - Formula 1 - Coppa america - Geometria per le scuole medie.

VIDEOTECA COMPUTER N. 6
per i possessori di Commodore 64

Nel manuale n. 6: LINGUAGGIO MACCHINA: 1ª lezione - Pseudocodice: I numeri a caso - La smazzata - Basic: 3ª lezione - Come tradurre un numero decimale alla base 2 o 16.

Nella cassetta n. 6: Buio - Istogrammi 3D - Black Jack - Mister Boa - Memo 101.

I numeri arretrati costano L. 15.000. Indirizzare vaglia o assegno a Editoriale VIDEO via Castelvetro 9 - 20154 Milano specificando il numero richiesto. Ufficio tecnico e arretrati: telefono 02/3184829

PLAY ON TAPE N. 1
per i possessori di VIC 20

Nel manuale n. 1: I tasti funzionali del VIC 20 - Pseudocodice e programmazione Basic - Il joystick.

Nella cassetta n. 1: Totocalcio - Air attack - Cervellone - Inferno 3D - Bilancio familiare.

PLAY ON TAPE N. 2

Nel manuale n. 2: Il basic più veloce - I caratteri speciali del VIC 20 - Disegnare con la tastiera e il joystick - Pseudocodice: 2ª lezione.

Nella cassetta n. 2: Test per misurare il Quoziente intellettuale - Easyword - Caccia al tesoro - Gestione Magazzino - Formula 1.

PLAY ON TAPE N. 3
per i possessori di VIC 20

Nel manuale n. 3: I cicli annidati del Basic - Come sviluppare un programma (Corsa automobilistica - Piranha) - L'interfaccia sconosciuta - Conosci il tuo VIC 20?

Nella cassetta n. 3: Sette e mezzo - Dieta - Calorie dei cibi - Gangster - Costi chilometrici.

PLAY ON TAPE N. 4
per i possessori di VIC 20

Nel manuale n. 4: Pseudocodice: Istruzioni read e data - Figura richiama - Grafici a barra. Il basic del VIC 20 - I linguaggi di programmazione - Come personalizzare i programmi.

Nella cassetta n. 4: G.O MO-KU - Calcolatrice - Golf - Vic Tab - Cabala e sogni.

PLAY ON TAPE N. 5
per i possessori di VIC 20

Nel manuale n. 5: Pseudocodice: 5ª lezione - I sistemi di numerazione - Le periferiche di ingresso e uscita - BASIC: I cicli.

Nella cassetta n. 5: Atterraggio - Memory Trainer - Stop Music - Program Recorder: 1) Crealista, 2) Stampalista - Istocambio.

PLAY ON TAPE N. 6
per i possessori di VIC 20, C.16, PLUS/4

Nel manuale n. 6: LINGUAGGIO MACHINA: 1ª lezione - Pseudocodice: I numeri a casa - La smazzata - Basic: 3ª lezione.

Nella cassetta n. 6:
Lato Vic 20: Redknight - Il risparmiatore - Api e ragni - Quindici 3D - Oroscopo.
Lato C.16 - Plus/4: Totocalcio - Colour Search - Roulette - Wargame - Bioritmo.

PLAY ON TAPE COMPUTER 8

VIC 20:

- METEORE
- THE LAST BATTLE
- SEVEN ELEVEN
- TELEFONO
- RIFLESSOMETRO

C. 16 - PLUS/4:

- RIFLESSOMETRO
- QUADRATI
- C.16 MEMORY
- CONT. TELEFONICO
- DAMA

EV EDITORIALE VIDEO