PLAY ON TAPE

COMPUTER

manuale

7

VIC 20
C. 16
PLUS/4

LINGUAGGIO MACCHINA:
2° LEZIONE

IL BASIC E
LE PERIFERICHE

PSEUDOCODICE:
- Liste legate
- Il bestiario

EV EDITORIALE VIDEO

Nella grande confusione dell'edicola dedicata all'informatica sventola distinguendosi la bandiera di PLAY ON TAPE Computer, il mensile con cassetta dedicato ai possessori di VIC 20, C.16 e Plus/4. I nostri lettori sanno che noi non consideriamo il computer come uno strumento per usare i videogiochi, a differenza della maggior parte delle pubblicazioni di settore che appaiono in edicola. Nelle nostre cassette ci sono anche i videogiochi, però nell'ambito di una raccolta di software che arricchisce e completa la macchina, che aiuta a comprenderne la logica e le possibilità.

Altrettanto utili alla macchina sono i nostri manuali che costituiscono una raccolta davvero indispensabile. Gli articoli e le lezioni non sono certo dedicati solo ai super specialisti, anche se questi ultimi li apprezzano molto, e di sicuro sono utili anche a coloro che, dopo aver cominciato a prender confidenza con il loro computer, vogliano conoscere i segreti della programmazione.

Nel manuale di questo mese appare la seconda lezione di Linguaggio Macchina, che completa la conoscenza del microprocessore, necessaria per poter approfondire l'argomento. Continuano le lezioni di Pseudocodice: la settima si occupa delle Liste legate. Delle periferiche, invece, si tratta nella terza lezione del Basic.

La cassetta di PLAY ON TAPE Computer n. 7 presenta, dal lato del Vic 20, un programma-gioco di Anagrammi, quindi Ponticello, un interessante Puzzle, il divertente Sonar Cave, e Julianus Dies, il programma che consente di cogliere un altro aspetto del calendario. Dal lato C.16 e Plus/4 c'è Hat in the ring, il programma di simulazione di una campagna elettorale alla presidenza degli Stati Uniti, quindi c'è Dieta, che consente di stabilire quale sarebbe il peso ideale per ciascuno di noi e quante calorie si dovrebbero ingerire quotidianamente per tornare in forma o per restarci. Il programma seguente stabilisce appunto quante calorie contengono i cibi e permette di organizzare una dieta personalissima, scientifica e fantasiosa.

La cassetta presenta quindi un divertente Poker da giocare contro il computer e un gioco del Quindici a 3 dimensioni.

Questi i contenuti del numero 7 che va ad accrescere la vostra collezione. Il prossimo mese vi riserveremo qualche altra simpatica sorpresa. Arrivederci presto.

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 Monteverdi, Fabio Macchioni, Tito Caponio, Sebastiano Pastore, Domenico Cellamare, Roberta Di Pietro, Alessandro Vallone, Fabio Rossi, Arnaldo Restelli, Dino Ticli, Igino Zaffaina, Giovanna Zampella - Stampa: Color Graf Milano - Fotocomposizione: ERREGI Milano - reg. del Trib. di Milano n. 518 del 10/11/84 - Stampato in Italia

• LISTE LEGATE
• IL "BESTIARIO"

Il lettore attento, che ha seguito la serie di articoli di pseudocodice, avrà senz'altro notato che lo sforzo del progetto si è sempre più spostato dall'ALGORITMO alla STRUTTURA DEI DATI. Questo fatto non è casuale, e spieghiamo subito il motivo.
L'ALGORITMO esprime la logica seguita dalla soluzione proposta dal programmatore, evidenziando le scelte (alternative), le ripetizioni e le trasformazioni elementari (istruzioni) necessarie per raggiungere lo scopo.
La STRUTTURA DEI DATI rappresenta invece la scelta del programmatore di scorgere una struttura interna nei dati a disposizione del programma. Tale struttura interna è così importante che sarà proprio da qui che si intuirà l'algoritmo. Il momento in cui riassumiamo la struttura dei dati in una delle organizzazioni semplici o complesse, è già il momento in cui cominciamo a scorgere una soluzione.
Alcuni dei metodi di progetto più diffusi, per esempio il "Metodo Jackson" fa di questa intuizione la parte centrale del proprio insegnamento: semplicemente impostando uno schema dei dati, abbiamo anche impostato un modo di risoluzione.
Senza voler arrivare a tanto, è senz'altro vero che ogni soluzione è già nascosta dentro ai dati in nostro possesso: si tratta di estrarla, così come lo scultore estrae la figura dalla creta senza forma.
La possibilità di organizzare i dati in modo da esprimerne i rapporti interni è perciò uno degli strumenti più potenti a disposizione del programmatore.
In questo articolo approfondiamo la nostra conoscenza delle strutture di dati complesse, introducendo il concetto di "legame" fra liste.

LEGAMI E PUNTATORI

Un PUNTATORE è una (qualunque) variabile numerica che contenga l'INDICE di una struttura dati semplice (vettore o matrice). In una lista per esempio, il puntatore al primo elemento libero della lista, è una variabile numerica che contiene l'indirizzo del primo elemento vuoto del vettore LISTA (vedi articolo 5).
Fino ad ora i puntatori erano esterni alle strutture dati complesse che abbiamo studiato: il puntatore di inizio e fine lista circola-

re, il puntatore all'elemento di testa di una coda e così via. In questo modo le nostre strutture complesse si potrebbero rappresentare come in figura 1 dove per rappresentare il puntatore abbiamo usato un quadratino con una freccia che indica l'elemento puntato. (Nella realtà la freccia è sostituita dall'indice contenuto in P).

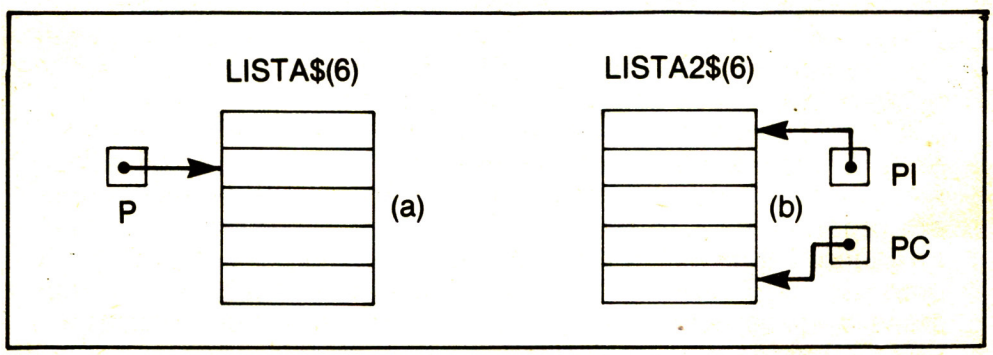


Fig. 1. Esempio di rappresentazione grafica di una lista (a), e di una lista circolare (b).

Le strutture dati complesse così descritte sono dette LINEARI, perché in fondo gli elementi si trovano "uno dietro l'altro", inseriti nei vettori di supporto e quindi non vi è necessità di informazioni aggiuntive per specificare dove si trova il prossimo elemento. Infatti nella stampa di una lista o di una coda è sufficiente sapere l'inizio e la fine dei dati e stampare tutti gli elementi compresi. Con le strutture lineari il programmatore ha la possibilità di esprimere legami lineari di dati: le carte contenute in un mazzo, la coda allo sportello di un ufficio, la lista degli appuntamenti, l'elenco telefonico personale (con aggiunte, cancellazioni e modifiche).
Ma se vogliamo invece rappresentare dei collegamenti tra dati raggruppati in un'unica struttura abbiamo la necessità di esprimere un RAPPORTO tra dati, che non è semplicemente la successione.
Facciamo un esempio concreto sperando di dare anche un'idea creativa ai nostri lettori.
Stiamo imparando i nomi degli animali in una lingua straniera, per esempio in inglese. Vorremmo memorizzarli in un elenco sul nostro fedele PC e nello stesso tempo tenerli sotto controllo in modo da poter ottenere elenchi per noi significativi, come:

- l'elenco di tutti i mammiferi (rettili, pesci, uccelli o anfibi)
- l'elenco delle "famigliole": il maschio, la femmina ed il piccolo (per esempio TORO, MUCCA e VITELLO).

La nostra idea è di far crescere il nostro "bestiario" nel tempo, aggiungendo ogni giorno qualche nome in più in modo che si agganci con tutti gli altri, senza la pretesa di introdurli tutti in una volta.
Ogni animale, perciò, è sia un elemento separato che una informazione che si incastra nel mosaico del bestiario già inserito. Per inciso, in inglese può essere molto utile costruire simili dizionari personali, per gli animali come per qualunque altro termine linguistico (verbi, oggetti, etc...): diventerà un modo divertente (e imprevedibile) di acquistare conoscenza.
Nel nostro esempio tratteremo il caso in cui le uniche informazioni di legame fra gli animali saranno la CLASSE (mammiferi, anfibi, rettili, pesci o uccelli) ed i SOTTOGRUPPI (maschio, piccolo o femmina). Con lo stesso sistema però potremo aggiungere qualificazioni più sottili, moltiplicando lo schema per tanti nuovi attributi quanti ne vorremo. Consigliamo di partire con un massimo di sette: già con 15 il nostro bestiario diventerebbe davvero un bel giardino zoologico!
Tornando al problema tecnico vediamo che abbiamo toccato il punto fondamentale che distingue una lista lineare da una lista con legami: noi vogliamo memorizzare non solo i dati, ma anche i rapporti che li uniscono. Nel nostro esempio, quando introduciamo il termine MUCCA, vogliamo che abbia una concatenazione con TORO e con VITELLO, in modo che in seguito si possa ricostruire la famigliola.
I legami fra i dati vengono memorizzati sotto forma di puntatori ad altri elementi della struttura. Questi puntatori, al contrario di quelli delle liste lineari, sono da considerare come parte integrante della struttura dati complessa.
Il nome di questa struttura è LISTA LEGATA, in quanto i vari elementi sono come concatenati da puntatori. (In inglese "LINKED LIST").
Un ELEMENTO di una lista legata è formato da 2 componenti:

a. l'informazione, che contiene il dato memorizzato con tutte le informazioni rilevanti.
b. l'insieme dei puntatori di legame, uno per ogni tipo di legame che si vuole stabilire.

Essendo tale lista una struttura dati complessa, per essere realizzata deve far ricorso a strutture dati elementari.
Componenti elementari:

a. l'informazione è contenuta in un vettore (variabile con indice) chiamato INFO$(m)
b. i puntatori di legame: poiché ve ne è una serie per ogni elemento INFO$, segue che essi sono delle variabili a doppio indice: PUN (i,m). Il primo indice indica il tipo di legame, ed il se-

condo indica a quale elemento si riferisce.

Nel caso del nostro bestiario inglese:

a. L'informazione relativa al nome dell'animale (che introduciamo, insieme alla traduzione, nel vettore INFO$).
b. I legami saranno: CLASSE e SUCCESSORE DI FAMIGLIA.

Ma cosa si memorizza esattamente per ognuno dei puntatori di legame?

Ecco la regola magica del legame:
ogni puntatore di legame punta all'elemento della lista legata che è suo SUCCESSORE NELL'ORDINE DI LEGAME.
Perciò nel nostro caso avremo che per ogni qualsiasi elemento K:

— PUN(1,K): punta al prossimo animale nella lista della stessa CLASSE: (al prossimo mammifero se INFO(K) è un mammifero, al prossimo pesce se INFO(K) è un pesce, e così via)
— PUN(2,K): punta all'animale nella lista legato da "parentela" (per esempio se INFO(K) è "GATTO", punta a GATTA o GATTINI, a seconda di quale animale è stato introdotto).

Esternamente alla lista legata, servono i puntatori ad ogni inizio di legame: ci vuole il puntatore al primo mammifero, al primo pesce, al primo uccello ed al primo anfibio della lista. (Nel nostro caso non serve un puntatore d'inizio della prima famigliola, perché ognuna di esse non è legata alle altre).
Ultima cosa da sapere, prima di esaminare gli algoritmi di gestione della lista, è che useremo il valore convenzionale 0 (zero),

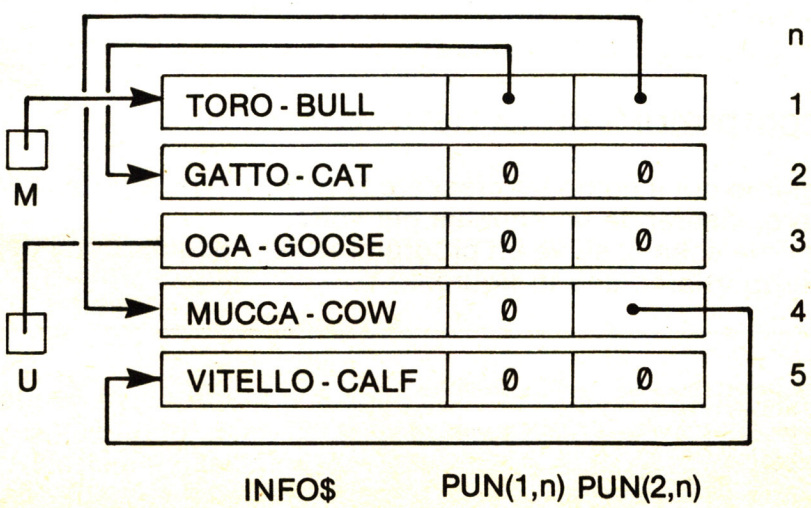


Fig. 2

per indicare il puntatore vuoto, cioè la fine di una catena di legame. Per scorrere tutti i mammiferi, per esempio, partiremo dal primo (puntato dal puntatore di inizio) e seguiremo i puntatori di legame PUN(1,) fino a trovare il puntatore vuoto.
Diamo ora un esempio grafico di una lista in cui abbiamo introdotto in questo ordine 5 animali: TORO, GATTO, OCA, MUCCA, VITELLO.

Nel disegno, M ed U sono i puntatori esterni al primo mammifero ed al primo uccello della lista rispettivamente. Come si vede, la lista contiene gli animali nell'ordine in cui sono stati introdotti, ma seguendo le frecce, si possono ricostruire le sequenze dei mammiferi (seguendo PUN(1,)) e la famigliola del TORO (seguendo PUN(2,)).
In questo altro disegno, sostituiamo alle frecce i numeri indice di elemento e avremo un'idea esatta del contenuto delle nostre variabili:

	TORO - BULL	2	4	1
1 M	GATTO - CAT	0	0	2
	OCA - GOOSE	0	0	3
3 U	MUCCA - COW	0	5	4
	VITELLO - CALF	0	0	5

Fig. 3

LA COSTRUZIONE DELLA LISTA LEGATA

Pensiamo ora a come far crescere la nostra lista legata a partire da zero, inserendo un animale per volta.
Per prima cosa ci serve un programma per l'acquisizione dei dati. Ne riportiamo uno in figura 4.

```
10 DIM INFO$(100),PUN(2,100),MUPRA(5)
20 LL=0:FDATI=100
30 INPUT "INSERZIONE (1),ESTRAZIONE (2),FINE (3) ?";CT
40 IF CT=1 THEN 50
41 IF CT=2 THEN 570
42 IF CT=3 THEN 700
50 PRINT"       FASE DI INTRODUZIONE DATI"
51 IF CT=2 THEN 570
```

```
60 INPUT "NOME ITALIANO DELL'ANIMALE ?";NI$
70 IF NI$="STOP" THEN 30
80 INPUT "NOME INGLESE DELL'ANIMALE ?";EN$
90 INPUT "MASCHIO (M),FEMMINA (F) O PICCOLO (P) ?";SE$
100 INPUT "MAMMIFERO(1),UCCELLO(2),PESCE(3),       RETTILE(4),ANFIBIO(5) ?";CL
110 INPUT "ANIMALE PARENTE ?";AP$
120 B$=" ":L1=8-LEN(NI$)
130 L2=8-LEN(EN$)
140 FOR A=1 TO L1:S1$=S1$+CHR$(32):NEXT
150 FOR A=1 TO L1:S2$=S2$+CHR$(32):NEXT
160 INF$=NI$+S1$+S2$+EN$+SE$
```

Fig. 4. Raccolta dei dati.

Riassumiamo le informazioni che il programma richiede:

—nome italiano dell'animale: NI$
—nome inglese dell'animale: EN$
—maschio, femmina o piccolo: SE$
—classe (mammifero, uccello, pesce, rettile o anfibio): CL
—animale parente: AP$

Notiamo che le informazioni NI$, EN$ ed SE$ vengono concatenate insieme nella variabile INF$ (istruzione 160). Il segno + è l'operatore della concatenazione alfanumerica di stringhe. Le istruzioni 120-150 servono per uniformare a 8 caratteri il nome italiano ed inglese, in modo che questi occupino in INF$ una posizione fissa.

```
170 REM ********************************
180 GOSUB 300:REM ** INSERISCI ANIMALE *
185 IF LEN(AP$)=0 THEN GOSUB 370:REM ********* AGGIORNA LEGAME DI CLASSE *****
190 IF LEN(AP$)<>0 THEN GOSUB 470:REM ******** AGGIORNA LEGAME DI PARENTELA **
200 IF FOUND=0 THEN 270
210 REM ********************************
220 PRINT "* ANIMALE "NI$" INSERITO *":PRINT
240 N=N+1:IF N<10 THEN 60
260 REM ********************************
270 PRINT"* ANIMALE PARENTE NON TROVATO *"
275 GOSUB 370:REM *** AGGIORNA LEGAME DI CLASSE ***
280 GOTO 210
```

Fig. 5. Richiamo delle funzioni di inserzione e aggiornamento dei legami.

In figura 5 è riportato il proseguo del programma, che provvede a inserire l'animale nella lista. Osservate che mentre l'inserzione di un animale è fissa, l'aggiornamento del legame di parentela è eseguito solo se non vi è animale parente: infatti nella nostra lista gli animali legati per parentela non sono legati anche per

classe (vedere per esempio la figura 2: MUCCA è raggiungibile solo attraverso TORO seguendo il legame di parentela, cioè PUN(2,1)).
Del resto le istruzioni 270-280 servono per aggiornare il legame di classe quando l'animale parente non è presente nella lista - cioè FOUND = O - in questo caso si tratta di un errore e bisogna impedire che l'animale risulti inaccessibile.
A questo punto siamo in grado di scrivere l'algoritmo per l'inserimento dell'animale. La prima cosa a cui pensare è una variabile LL che indichi l'ultima locazione occupata della lista: inizialmente LL varrà zero.
Abbiamo quindi:

```
INIZIO Inserisci animale
 LL = LL + 1
 SE LL > FLISTA ALLORA STOP
 INFO$(LL) = INF$
 PUN(1,LL) = O; PUN(2,LL) = O
FINE
```

L'inserzione è semplice: si assegna a INF$ il suo posto nella lista aggiungendo 1 a LL, e controllando che LL non sia superiore alla lunghezza complessiva di INFO$, cioè FLISTA.
Consideriamo ora l'aggiornamento del legame di classe: si tratta di esaminare tutta la catena degli animali della stessa classe, trovare l'ultimo (che avrà 0 come PUN(1,n)) e modificare PUN(1,n) in modo da puntare all'elemento appena inserito - cioè il contenuto di LL.
I cinque puntatori di classe (Mammiferi, Uccelli, Pesci etc...) sono in un vettore MUPRA(5) di 5 elementi; la variabile CL ci indica quale utilizzare per trovare l'inizio di ogni classe.

```
INIZIO. Aggiorna legame di classe.
 SE PI = O ALLORA MUPRA (CL) = LL
 ALTRIMENTI
  RIPETI
  FINCHÉ PUN(1,PI) ≠ O
   PI = PUN(1,PI)
  FINE_RIPETI
  PUN(1,PI) = LL
 FINE_SE
FINE.
```

Il "SE" serve per controllare se ancora non abbiamo inserito alcun animale di quella classe: in questo caso si inizializza il puntatore MUPRA appropriato.
Altrimenti si SEGUONO le "frecce": si scorre la lista legata - (ciclo RIPETI) finché non si individua l'elemento il cui puntatore PUN(1,n) vale 0.

A questo punto aggiorniamo questo puntatore col contenuto di LL, cioè il puntatore all'elemento nuovo appena aggiunto. Il programma si trova in fig. 6.
Seguiamo il procedimento dell'introduzione di animali: TORO, GATTO, MUCCA, per ciò che riguarda il legame di classe.
MUCCA ha anche un legame di parentela con TORO: dobbiamo quindi aggiornare il legame di parentela di TORO, cioè PUN(2,1).
Per trovare TORO nella lista potremmo: 1) esaminare INFO$ dall'inizio fino a LL; 2) seguire il legame di mammifero dall'inizio fino al puntatore Ø.
Nel primo caso prenderemmo in considerazione il vettore INFO$ come se fosse una struttura elementare, e passeremmo in rassegna tutti gli animali, senza distinzione di classe. Nel secondo caso esamineremmo la struttura lungo i percorsi della lista legata, e passeremmo quindi in rassegna solo gli animali della stessa classe.
Entrambe le soluzioni sono possibili, con preferenza per la soluzione 2, che impedisce di "imparentare" due animali di classe diversa: non rischieremmo così di imparentare un SERPENTE con un CAVALLO!
La soluzione n° 1, comunque è in tutto e per tutto identica alla ripetizione del programma "Aggiorna legame di classe" e perciò si lascia al lettore il compito di realizzarla per conto proprio.
Riportiamo qui, invece, la soluzione 1, in modo anche da evidenziare il fatto che INFO$ può essere visto anche come struttura indipendente, e gestita come un semplice vettore:

```
INIZIO. Aggiorna legame di parentela
 J = 1
 RIPETI
 FINCHÉ (J < LL) AND (AP$ ≠ INFO$(J))
  J = J + 1
 FINE_RIPETI
 SE INFO$(J) = AP$
 ALLORA
  SE PUN(2,J) ≠ O
  ALLORA
   J = PUN(2,J)
  FINE_SE
  PUN(2,J) = LL
 ALTRIMENTI
  FOUND = Ø "Animale non trovato"
FINE_SE
```

Notate che nel programma, una volta trovato l'animale parente, bisogna controllare se il suo puntatore di parentela PUN(2,J) è già impostato. In quel caso si scorre il legame di un elemento e

poi si aggiorna il legame di parentela. In figura 6 il programma BASIC traduce questo algoritmo.

```
290 REM ********************************
300 REM ******* INSERZIONE ANIMALE
310 LL=LL+1
320 IF LL>FDATI THEN 780
330 INFO$(LL)=INF$
340 PUN (1,LL)=0:PUN (2,LL)=0
350 RETURN
360 REM ********************************
370 REM *** AGGIORNA LEGAME DI CLASSE ***
380 PI=MUPRA(CL)
390 IF PI=0 THEN MUPRA(CL)=LL:GOTO 450
400 IF PUN (1,P)=0 THEN 440
410 PI=PUN(1,PI)
420 GOTO 410
430 PUN(1,PI)=LL
440 RETURN
450 RETURN
460 REM ********************************
470 REM *** AGGIORNA LEGAME DI PARENTELA **
480 L3=LEN(AP$)
490 FOR J=1 TO LL
500 IF AP$=MID$(INFO$(J),1,L3) THEN 520
510 NEXT J
520 IF MID$(INFO$(J),1,L3)<>AP$ THEN FOUND=0:RETURN
530 IF PUN (2,J)<>0 THEN J=PUN (2,J)
540 PUN (2,J)=LL:FOUND=1
550 RETURN
```

Fig. 6. Inserzione e aggiornamento dei legami.

LA FASE DI ESTRAZIONE

Fino ad ora abbiamo visto come gestire la parte di introduzione di dati nella lista legata. Questa si chiama fase di "UPDATE" (cioè aggiornamento).
Ci restano ora da realizzare gli algoritmi che ci permettono di rivedere le nostre informazioni. Questa parte si chiama fase di "INQUIRY" (in italiano "interrogazione"), e in generale quindi i programmi che, come il nostro, archiviano e interrogano i dati si chiamano programmi di "inquiry-update".
L'interrogazione è una funzione importante per trarre il massimo dalle informazioni memorizzate: maggiore è la libertà di fare "inquiries" complesse, maggiore è l'impressione di potenza che il nostro programma ci trasmette.
Una "inquiry" si divide in 2 parti: un criterio di scelta degli elementi da esaminare, e il tipo di operazione che si vuole eseguire. Per esempio, sull'elenco di una rivista di motori l'interrogazione

"Quali automobili FORD sono offerte?" si può distinguere in:

1. La scelta delle sole automobili FORD
2. L'elencazione di tutti i modelli.

Nei programmi più sofisticati, il criterio di scelta può essere espresso in modo molto complesso, e le operazioni possono essere diverse (p.e.: mettere in ordine, elencare, sommare etc...).
Tornando al nostro esempio ci limitiamo a fornire la possibilità di scegliere una delle cinque classi di animali: la funzione è sempre quella di elencare gli animali introdotti evidenziando però l'ordine di parentela.
Per scorrere la lista secondo l'ordine di parentela, ma scegliendo però tutti gli animali di una stessa classe, occorre gestire appositamente PUN(1,n) e PUN(2,n). Si tratta di scendere lungo PUN(1,n) finché non troviamo un PUN(2,n) diverso da 0. A questo punto scorriamo il legame di parentela fino a esaurimento. Ritorniamo poi a seguire PUN(1,n).
L'algoritmo si presenta perciò come due "RIPETI FINCHÉ" annidati:

```
INIZIO. Scansione lista seguendo i legami
  PC = primo puntatore della lista di classe
  Stampa "INIZIO lista"
  RIPETI
  FINCHÉ PC ≠ O
    Stampa INFO$(PC)
    PF = PUN(2,PC) "Imposta puntatore di parentela"
    RIPETI
    FINCHÉ PF ≠ O
      Stampa INFO$(PF)
      PF = PUN(2,PF)
    FINE_RIPETI
    Stampa "fine famigliola"
    PC = PUN(1,PC)
  FINE RIPETI
  Stampa "Fine lista"
FINE
```

```
560 REM ********************************
570 PRINT "          FASE DI INQUIRY"
580 PRINT"SELEZIONE:   "
590 INPUT "MAMMIFERO(1),UCCELLO(2),PESCE(3),      RETTILE(4),ANFIBIO(5) ?";CL
600 PC=MUPRA(CL)
605 PRINT"INIZIO LISTA"
610 IF PC=0 GOTO 685
615 PRINT" ";INFO$(PC);
620 PF=PUN(2,PC)
630 IF PF=0 THEN 671
640 PRINT"--"INFO$(PF);
650 PF=PUN(2,PF)
```

```
671 PRINT"**"
675 PC=PUN(1,PC)
680 GOTO 610
685 PRINT" FINE DELLA LISTA "
688 REM ***********************************
690 GOTO 570
695 REM ***********************************
700 STOP
730 PRINT "FINE LISTA LIBERA ":STOP
```

Fig. 7. Così si scorre la lista seguendo i vari legami.

In figura 7 si vede la realizzazione di questo algoritmo.

```
 Nome italiano dell'animale ?? STOP
 Inserzione(1)  o Estrazione(2) o Fine(3) ?? 2
        FASE DI INQUIRY
Selezione :
  Mammifero(1) Uccello(2) Pesce(3) Rettile(4) Anfibio(5) ?? 1
Inizio della lista
  mucca    cow      f -- toro     bull     m **
  cane     dog      m -- cagna    bitch    f **
Fine della lista
        FASE DI INQUIRY
Selezione :
  Mammifero(1) Uccello(2) Pesce(3) Rettile(4) Anfibio(5) ?? 2
Inizio della lista
  oca      goose    f **
Fine della lista
        FASE DI INQUIRY
Selezione :
  Mammifero(1) Uccello(2) Pesce(3) Rettile(4) Anfibio(5) ?? 4
Inizio della lista
  vipera   adder    m **
Fine della lista
        FASE DI INQUIRY
Selezione :
  Mammifero(1) Uccello(2) Pesce(3) Rettile(4) Anfibio(5) ??
```

Fig. 8. Esempio di INQUIRIES.

Nel prossimo numero esamineremo il concetto di FILE unito a quello di LISTA LEGATA per ottenere la gestione di un FILE ad INDICE.

M.C.

MICROPROCESSORI: 2ª PARTE

Questa volta ci occupiamo del funzionamento interno di un UP, della sua architettura, dell'assembler e degli indirizzamenti per potere così fornire le basi necessarie ed utili per una migliore comprensione del L/M, argomento che affronteremo nella prossima puntata.

2.1 ARCHITETTURA INTERNA DI UNA CPU.

Addentriamoci ora nella parte più interna delle CPU. In ognuna di esse esistono sempre delle celle di memoria indipendenti da quelle pilotate esternamente dall'"Address Bus". Tali locazioni prendono il nome di "registri" e possono essere utilizzate per varie funzioni oppure avere un solo ruolo specifico. La loro dimensione (la capacità di memorizzare un numero di bits), può variare da CPU a CPU, ed in una sola CPU da registro a registro. Esistono comunque due tipi di CPU, a seconda del loro set di registri: quelle "memory oriented" (6510 per esempio) in cui le operazioni sono quasi sempre basate su dati della memoria esterna (pilotata dall'Address Bus) ed in cui non esistono grandi registri, e quelle "register oriented" (Z-80), che posseggono un ampio set di registri interni che provvede a sostituire la memoria esterna nelle varie operazioni. Comunque, a prescindere dall'appartenenza di una CPU ad una famiglia piuttosto che ad un'altra, esistono alcuni tipi di registri comuni a quasi tutti:

il "PROGRAM COUNTER" (PC), un registro che ha l'unico compito di tenere memorizzato il valore della locazione di memoria con cui il "Data Bus" è in comunicazione. In pratica il contenuto di questo registro è quello che viene trasmesso sull'"Address Bus" per selezionare una data cella di memoria;

lo "STACK POINTER" (SP), il cui contenuto costituisce l'indirizzo di alcune celle di memoria ad esclusivo uso della CPU;

lo "STATUS REGISTER" (SR), il cui contenuto è un'indicazione del risultato di un'operazione eseguita dalla CPU.

Di questi registri, che hanno solo i particolari ruoli citati, parleremo accuratamente più avanti. Un altro registro comune a quasi tutte le CPU è:

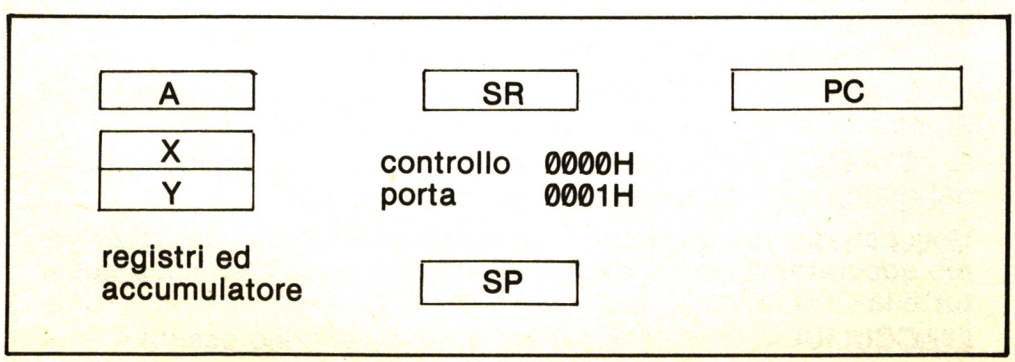
l'"ACCUMULATORE" (A), nel cui interno vengono eseguite ope-

razioni logiche, aritmetiche ed altre istruzioni speciali.

Esistono inoltre registri che possono assumere vari ruoli e il cui nome varia a seconda della CPU a cui appartengono.

2.2 ARCHITETTURA INTERNA DELLA 6510

Innanzitutto precisiamo che chi possiede un computer con all'interno una CPU 6502 (VIC 20) può seguire il presente corso senza alcun problema, in quanto tale CPU è equivalente ad una 6510. Quest'ultima è una delle più tipiche CPU "memory oriented" per cui la sua struttura interna ed il suo set di registri si rivela alquanto semplice. Innanzitutto è dotata di un PROGRAM COUNTER di 16 bits, che le permette di indirizzare fino a 64K di RAM. Lo STACK POINTER è di soli 8 bits, per cui può puntare in un campo di soli 256 bytes. Esistono inoltre un ACCUMULATORE ed uno STATUS REGISTER anche esso di 8 bits. Oltre a questi registri ordinari ne troviamo altri due speciali in cui è possibile memorizzare dati ed usarli per riferirsi a locazioni di memoria. Tali registri sono Y ed X, entrambi della dimensione di 8 bits. A questo punto si può capire che avendo a disposizione solo tre registri (A,X,Y) per la programmazione, molti dati che vorremo utilizzare li dovremo porre in locazioni di memoria esterna alla CPU per poterli poi richiamare. La CPU, oltre che un set di registri interno, possiede anche una "porta" che le permette di comunicare con una periferica che non sia la memoria. Questa "porta" ha la funzione di raccogliere e trasmettere dati tra CPU e periferiche (quali stampanti, registratori, floppy disks e così via). Tale "porta" corrisponde alla locazione di memoria 0001H, per cui se vorremo scrivere un dato in questa locazione non lo invieremo alla memoria ma alla "porta". Esiste poi la locazione di memoria 0000H che non fa altro che controllare la "porta" in 0001H in un modo di cui parleremo successivamente. Diamo adesso uno schema visuale dell'architettura interna della CPU 6510.



2.3 CODICI MNEMONICI

Abbiamo visto nel paragrafo 1.5 che una CPU accoglie un'istruzione sotto forma di codice numerico, di solito raffigurata in notazione esadecimale, e che ogni codice può rappresentare indifferentemente un dato su cui lavorare o un'istruzione da eseguire. A questo punto sarebbe impossibile per un essere umano ricordare a memoria tutti i codici corrispondenti alle istruzioni (lo Z-80 ne possiede oltre 600) da usare in un programma. Allo scopo di rendere più semplice ed immediata la programmazione si usano i codici mnemonici, così detti appunto perché più facili da capire e da ricordare. Tali codici sono delle vere e proprie parole che rappresentano all'operatore in modo rapido la funzione dell'istruzione da eseguire. La traduzione di un codice mnemonico in codice numerico interpretabile dalla CPU, spetta poi ad un programma esterno chiamato "Assembler" (assemblatore) nel caso si lavori con esso; oppure si usano determinate tabelle che mostrano la corrispondenza numerica dei codici mnemonici. Si usa distinguere la programmazione in l/m con codici numerici da quella in Assembler con codici mnemonici, ma il risultato è comunque lo stesso: quello di comporre una serie di codici che vanno a formare un programma in l/m.

2.4 INDIRIZZAMENTI

Introdotta l'istruzione mnemonica, possiamo notare che una istruzione si riferisce quasi sempre ad un dato che la CPU deve prendere, elaborare e di cui in seguito deve memorizzare il risultato. In linguaggio macchina tutti i dati e le istruzioni sono contenuti nella RAM esterna alla CPU, la quale accede alle istruzioni leggendo una serie di celle consecutive che contengono i vari bytes rappresentanti istruzioni o dati. Importante ruolo svolge quindi il PROGRAM COUNTER, il quale tiene memorizzato l'indirizzo dell'istruzione che la CPU sta leggendo e che dopo ogni lettura di istruzione viene incrementato per puntare alla prossima istruzione da eseguire. Quindi notiamo che per l'esecuzione delle istruzioni il PC ha in sé tutte le informazioni necessarie. Diverso è invece lo specificare una locazione di memoria su cui eseguire un'operazione in quanto, se per esempio vogliamo caricare un dato numerico in un registro, il PC provvede a far leggere l'istruzione di caricamento, mentre la cella di memoria da cui caricare deve essere specificata nell'istruzione stessa. Esistono molti modi quindi di riferirsi ad un dato su cui operare: l'insieme dei modi viene definito "indirizzamento".

2.5 CODICI MNEMONICI 6510

In un'istruzione Assembler per 6510 si riconoscono fondamentalmente tre parti: oltre l'Op code, che costituisce la traduzione in l/m del codice mnemonico, esistono uno Mnemonic code (MN code), che rappresenta il tipo di operazione da eseguire, ed un Source Address, che rappresenta l'indirizzamento relativo al dato da usare nell'istruzione. Per esempio:

Op code	MN code	S.A.
AD_{16} 00_{16} 01_{16}	LDA	$1000

Tale istruzione "dice" alla CPU di riempire l'ACCUMULATORE (A) col contenuto della memoria 1000H. Lo MN code rappresenta l'istruzione Load A (carica l'ACCUMULATORE) mentre il S.A. fornisce informazioni riguardo al reperimento del dato da inserire nell'accumulatore (cella di memoria numero 1000H). La "$" davanti al 1000 significa solo che l'indirizzo è dato in notazione esadecimale.

2.6 I TIPI DI ADDRESSING

Addressing è un termine inglese per definire "indirizzamento" ed è una parola comune a tutta la letteratura riguardante CPU e computers. In particolare in una CPU 6510 esistono vari tipi di "Addressing":

IMMEDIATE ADDRESSING (Indirizzamento immediato)
Un'istruzione che utilizzi questo tipo di indirizzamento ha un codice formato da due bytes. Per esempio se noi vogliamo caricare l'ACCUMULATORE (A) con il numero 10H otterremo l'istruzione [LDA #$10].

Op code	MN code	S.A.
AD_{16} 10_{16}	LDA	#$10

Notiamo che l'istruzione LDA specifica che deve venire caricato nell'ACCUMULATORE il dato 10H che in Assembler 6510 si rappresenta in esadecimale come $10. Il simbolo "#" davanti al $10 indica all'Assembler che deve essere caricato il dato presente nel byte immediatamente successivo all'OP code, e non il dato presente nella cella di memoria numero 10H.

ZERO PAGE ADDRESSING (Indirizzamento a pagina zero)
Questo indirizzamento serve per indicare che il dato da caricare si trova nei primi 256 bytes di memoria, per cui viene fornito alla CPU un indirizzo formato da un solo byte che può appunto rap-

presentare una di tali locazioni. Se volessimo riempire A col contenuto della cella di memoria numero 1AH avremmo [LDA $1A].

Op code	MN code	S.A.
$A5_{16}$ $1A_{16}$	LDA	$10

Si noti subito che nel Source Address non è presente il simbolo "#", per cui il numero seguente rappresenta un indirizzo di memoria da cui prendere un dato e non il dato stesso. In questo indirizzamento il S.A. è la locazione di memoria 10H, il cui contenuto verrà trasferito nell'ACCUMULATORE.

ABSOLUTE ADDRESSING (Indirizzamento assoluto)
L'indirizzamento assoluto permette di trovare un dato in una qualsiasi locazione di memoria delle 65536 disponibili, per cui l'indirizzo da fornire alla CPU nell'istruzione, sarà di 16 bits (capace di indirizzare 65536 locazioni) e non di un byte come in pagina zero. La precedente istruzione in indirizzamento assoluto, ed usando come dato il contenuto della memoria 3400H, diventa [LDA $3400].

Op code	MN code	S.A.
AD_{16} 00_{16} 34_{16}	LDA	$3400

Il numero 3400H rappresenta l'indirizzo della cella di memoria da cui prendere il dato da trasferire. Il formato Assembler, notiamo, è il medesimo dell'indirizzamento a pagina zero; cambia l'Op code. È infatti compito del programma Assembler capire se un indirizzo può essere assoluto oppure in pagina zero (per l'operatore la forma di scrittura è la stessa).
Bisogna notare che dopo l'Op code vero e proprio (ADH) che rappresenta il LoaD assoluto, l'indirizzamento a 16 bits (3400H) è scomposto in due bytes (34H e 00H), che seguono l'Op code in modo che primo sia il byte di ordine minore (00H) e secondo quello di ordine maggiore (34 H). Tale tipo di ordinamento è comune a quasi tutte le CPU per cui, quando si memorizza un numero da 16 bits in due locazioni di memoria adiacenti da 8 bits, viene sempre memorizzato il byte di ordine minore nella locazione che viene esaminata per prima ed in quella immediatamente successiva il byte di ordine maggiore.

INDEXED ADDRESSING (Indirizzamento indicizzato)
In questo tipo di indirizzamento si fornisce alla CPU un indirizzo chiamato "base" che non rappresenta l'indirizzo reale da cui prelevare il dato necessario all'operazione. L'indirizzo reale viene ottenuto sommando alla base il contenuto di uno dei due registri indice X o Y. Se vogliamo caricare A col contenuto della memoria 3007H, possiamo caricare il registro X (istruzione LDX) con il

numero 07H e caricare in indirizzamento indicizzato fornendo una base di 3000H. Per cui otterremo il seguente programma:

```
LDX #$07        ;carica X col numero 07H
LDA $3000,X     ;carica l'accumulatore con 3000H indicizzato X
```

Op code	MN code	S.A.
BD_{16} 00_{16} 30_{16}	LDA	$3000,X

In Assembler l'Addressing $3000,X significa: se usiamo il dato contenuto nella memoria indicata dal numero ottenuto sommando a 3000H (la base) il contenuto del registro X (07H), otteniamo come indirizzo 3000H + 07H = 3007H. Questo tipo di indirizzamento può utilizzare indifferentemente il registro X o Y, come già detto, per cui avremmo avuto lo stesso risultato caricando Y col numero 07H e dando come Source Address il formato $3000,Y.

INDIRIZZAMENTI INDIRETTI

Ci accingiamo ora a parlare degli indirizzamenti indiretti, i più complessi usati dalla 6510. Il termine "indiretto" specifica che l'indirizzo di un dato da usare in un'operazione non viene inviato alla CPU insieme all'Op code, ma è memorizzato in due locazioni di memoria adiacenti sotto forma di due bytes (essendo l'indirizzo a 16 bits) con quello di ordine minore che precede quello di ordine maggiore. Noi quindi forniremo alla CPU l'indirizzo di queste due locazioni di memoria contenenti l'indirizzo reale, che verrà poi usato come puntatore del dato da usare nell'istruzione. Per fare un esempio possiamo inserire nelle locazioni di memoria 0002H e 0003H rispettivamente i numeri 30H e 17H. Se vogliamo usare in una operazione un dato che si trova nella locazione 1730H, tramite indirizzamento indiretto invieremo alla CPU l'indirizzo della locazione 0002H. Questa esaminerà che dato vi è nella locazione (30H) ed in quella successiva (17H), e accoppierà tali dati numerici formando l'indirizzo 1730H, che verrà usato per ricavare il dato da usare nell'istruzione. Il tutto schematizzato visivamente:

Indirizzo inviato alla CPU	Contenuto delle locazioni indirizzate	Indirizzo usato nella operazione
0002H	30H	
0003H	17H	1730H

INDIRECT INDEXED ADDRESSING
(Indirizzamento indicizzato indiretto)

Questo tipo di Addressing è possibile solo usando il registro Y.

Abbiamo già visto come l'indirizzamento indicizzato sia fondamentalmente formato da un indirizzo base sommato al contenuto di un registro (in questo caso Y). Nell'indirizzamento indicizzato indiretto il rapporto tra base e registro è identico, cambia solo il modo in cui la CPU ricava la base. Essa è infatti contenuta in due locazioni di memoria adiacenti e nell'Op code viene fornito alla CPU l'indirizzo di tali locazioni. La CPU, una volta ricavatasi la base, vi sommerà il contenuto del registro Y, ottenendo così l'indirizzo reale della cella di memoria il cui contenuto è da utilizzare. Ad esempio: riempiamo le locazioni 0002H e 0003H (N.B. qualsiasi locazione va bene) con i numero 70H e 53H rispettivamente ed il registro Y con il numero 12H. Usando l'indirizzamento indicizzato indiretto potremo scrivere [LDA ($0002),Y].

Op code	MN code	S.A.
$B1_{16}$ 00_{16} 02_{16}	LDA	($0002),Y

A questo punto la CPU esaminerà il contenuto delle locazione 0002H e 0003H (0002H + 01H) ottenendo l'indirizzo base 5370H a cui verrà poi sommato il contenuto del registro Y (12H) ottenendo 5370H + 12H = 5382H.
L'accumulatore sarà così caricato col contenuto della locazione di memoria 5382H.

INDEXED INDIRECT ADDRESSING
(Indirizzamento indiretto indicizzato)
Questo indirizzamento permette l'esclusivo uso del registro X. A differenza del precedente "indiretto", la somma tra base (data stavolta nell'Op code) e registro X, darà l'indirizzo delle locazioni di memoria il cui contenuto è l'indirizzo reale. In poche parole, mentre prima la CPU aveva un indirizzo A, da cui ricavare un indirizzo B per sommargli Y, qui la CPU somma all'indirizzo A il registro X: la somma indicherà dove trovare l'indirizzo B. Riempiamo le due locazioni di memoria 26H e 27H con i numeri 00H e 60H rispettivamente e il registro X con 06H. Se vogliamo riempire il registro A con il contenuto della locazione di memoria puntata dai bytes 26H e 27H scriveremo [LDA ($0020H,X)].

Op code	MN code	S.A.
$A1_{16}$ 20_{16} 00_{16}	LDA	($0020,X)

La CPU calcolerà la somma tra l'indirizzo 0020H ed il contenuto 06H del registro X, ottenendo 0020H + 06H = 0026H. Leggerà quindi i contenuti delle locazioni 0026H e 0027H e accoppiandoli otterrà l'indirizzo 6000H. L'accumulatore verrà caricato col contenuto della locazione di memoria 6000H.

RELATIVE ADDRESSING (Indirizzamento relativo)
Questo Addressing è un po' più complicato degli altri e verrà discusso in altre puntate esaminando le istruzioni di branch.

IMPLIED ADDRESSING (Indirizzamento implicito)
Questo indirizzamento è usato in istruzioni che specificano già in sè il S.A.. Se per esempio vogliamo incrementare il contenuto del registro X, scriveremo [INX].

Op code	MN code	S.A.
$E8_{16}$	INX	

Notiamo che non esiste un S.A., in quanto la CPU riconosce che nell'istruzione è già specificato dove si trova il numero da incrementare (in X) e non bisogna quindi dare nessun altro indirizzamento.
Per il momento chiudiamo questa seconda parte sperando di essere stati abbastanza chiari. Nel prossimo appuntamento parleremo delle vere e proprie istruzioni del l/m che, se avrete già appreso i "modi di indirizzamento", risulteranno abbastanza semplificate.

A.V.S.

IL BASIC E LE PERIFERICHE

Nei numeri scorsi abbiamo visto la struttura e l'impiego del linguaggio BASIC in relazione alle diverse esigenze di programmazione. Questa volta ci dedichiamo invece ad un uso spesso poco conosciuto, ma comunque molto interessante: la gestione delle periferiche.
Innanzitutto è bene chiarire cosa si intende con questo termine; in realtà, qualunque organo che consenta un "dialogo" fra noi e la macchina può essere considerato una periferica. La tastiera, mediante la quale noi comunichiamo con il computer, è una periferica, il video su cui vengono visualizzati i dati che ci interessano è anch'esso una periferica come lo sono la stampante, il registratore e il drive per floppy-disk.

Naturalmente un linguaggio di programmazione deve includere una serie di istruzioni dedicate al controllo di tutte le periferiche e il BASIC non fa eccezione.
Alcune di queste istruzioni le abbiamo già esaminate in precedenza (SAVE, LOAD, GET, INPUT ecc.) e sappiamo quindi usarle abbastanza bene; per le altre, invece, è necessario un breve discorso introduttivo.
Per poter comunicare con le periferiche il computer fa uso di "canali" ad esse collegati ai quali vengono poi associati dei "flussi" di dati. Naturalmente prima di inviare dati ad una periferica (ad esempio la stampante) bisogna comunicare al computer quale canale si desidera utilizzare e su quale flusso inviare i dati; ciò è possibile con la semplice istruzione OPEN seguita dal numero di file e da quello della periferica che si intende usare. Quindi se noi vogliamo aprire un file per scrivere dei dati sul registratore a nastro possiamo procedere come segue:

```
OPEN 1,1,1
```

in questo modo apriamo il file numero 1 sul canale 1 (che è appunto quello del registratore). Adesso possiamo scrivere dei dati come facevamo in precedenza, con dei semplici PRINT, ma specificando il numero di file su cui i dati andranno dirottati. Vediamo un esempio.

```
10 OPEN 1,1,1
20 INPUT A
30 PRINT # 1,A
40 IF A=0 THEN CLOSE 4:
   END
50 GOTO 20
```

Come probabilmente avrete capito, questo programma apre innanzitutto un file sul canale 1 (registratore), quindi chiede

all'operatore un dato numerico e successivamente lo salva su nastro inviandolo appunto al file numero 1. Se il numero inserito è zero, la linea 40 provvede a chiudere i files precedentemente aperti e conclude il programma.
Nell'esempio avrete notato l'uso di una nuova istruzione: CLOSE, che serve per chiudere i files aperti tramite l'apposita istruzione OPEN. Normalmente dopo CLOSE bisogna specificare il numero del file che si intende chiudere, in caso contrario verranno chiusi tutti i files aperti in quel momento. A qualcuno potrà venire spontaneo chiedersi lo scopo dell'istruzione CLOSE; dovete però ricordare che non è possibile aprire più di un certo numero di files contemporaneamente e inoltre, che ogni file aperto occupa un certo spazio in memoria che viene usato come "magazzino" temporaneo per i dati in arrivo o in partenza per una periferica. Per questi e altri motivi è sempre bene ricordarsi di chiudere un file quando si finisce di utilizzarlo per un particolare compito.
Ora che abbiamo imparato alcuni concetti fondamentali possiamo approfondire la nostra conoscenza di una periferica largamente diffusa presso tutti gli utenti di personal computer: la stampante, per la quale valgono le considerazioni generali fatte in precedenza. Quindi per inviare dati alla stampante dobbiamo prima aprire un file, stavolta sul canale 4, e quindi procedere come al solito con l'istruzione PRINT# seguita dal numero di flusso usato per inviare i dati.
Vediamo come al solito un breve esempio.

```
10 OPEN 1,4
20 PRINT#1,
   "PROVA STAMPANTE"
30 PRINT "SCHERMO"
40 CLOSE 4
```

Certamente avrete capito che questo programma, dopo aver aperto un file su stampante, stampa una frase su quest'ultima inviandola sul file 1 e successivamente esegue una normale stampa sul video. Capito la differenza?
In questo modo potrete mandare alla stampante i dati che vengono diretti sui files connessi al canale 4 che, come già detto, in condizioni normali è indirizzato alla stampante. Possono però verificarsi dei casi in cui desideriamo che tutti i dati normalmente visualizzati sul video siano dirottati verso la stampante; in questo caso possiamo usare l'utilissima istruzione CMD, la quale, se non diversamente specificato, invia tutti i dati sul numero di flusso indicato nell'istruzione stessa. Ad esempio, il seguente programma stampa sul video una serie di numeri ordinati

```
10 FOR A=1 TO20
20 PRINT A
30 NEXT A
```

sarà sufficiente inserire come prima linea la seguente serie di istruzioni per ottenere che tutti

i dati siano stampati su carta invece che sul video: OPEN 1,4: CMD 1.
Naturalmente la stampante è una periferica destinata, come il video, solo all'output, cioè all'emissione dei dati, ma vi sono altre periferiche, come il registratore o il drive, con cui si possono inviare o ricevere dati, naturalmente specificando prima il tipo di operazione che si desidera compiere.
Per il momento ci occuperemo esclusivamente del registratore a nastro che rimane comunque la periferica più usata benché anche il drive si stia diffondendo in maniera sorprendente (ma di questo discuteremo più a fondo successivamente).
Come abbiamo detto il registratore è una periferica che permette di memorizzare a tempo indeterminato dati o programmi; per quanto riguarda i secondi ne abbiamo già discusso nella lezione precedente: vediamo quindi come usare il registratore per archiviare piccole o medie quantità di dati che potranno essere successivamente riletti ed elaborati. Abbiamo già visto in precedenza dei semplicissimi esempi di uso del registratore per archiviare i dati: vediamo ora come è possibile rileggere questi dati.

```
 5 OPEN 1,1,1
10 INPUT"NOME ";A$
15 IF A$ = "" THEN GOTO 30
20 PRINT#1,A$ N = N + 1
25 GOTO 10
30 FOR S = 1 TO N
35 INPUT#1,A$
40 PRINT A$
50 NEXT S
60 CLOSE 1:END
```

A questo punto avrete sicuramente capito che per rileggere dei dati precedentemente salvati su nastro occorre usare l'istruzione INPUT# seguita dal numero di file su cui inviare i dati (in questo caso il registratore); ma esaminiamo in particolare questo programma. La prima linea apre un canale di comunicazione con il registratore; poi verranno chiesti all'operatore una serie di nomi che saranno successivamente salvati; l'operazione di inserimento continuerà finché non sarà introdotta una stringa nulla, nel qual caso il programma salterà alla linea 30 dove inizia un ciclo di lettura che stampa sul video tutti i dati. La variabile N tiene il conto dei dati inseriti.
Oltre ad INPUT esiste anche un'altra utile istruzione per la lettura di dati su nastro: si tratta di GET, che, a differenza di INPUT, preleva un solo carattere alla volta e non un'intera stringa, e che viene usata soprattutto per controllare singolarmente i dati presenti su nastro con un programma di questo tipo:

```
10 OPEN 1,1,1
20 GET#1,A$
25 C = ASC(A$)
30 PRINT C
40 GOTO 20
```

in questo modo i caratteri letti saranno di volta in volta convertiti nei corrispondenti codici

numerici e successivamente visualizzati sullo schermo.
Prima di terminare vediamo una breve tabella riassuntiva dei canali e delle periferiche ad essi collegate.

REGISTRATORE	1
MODEM	2
STAMPANTI	4/5
DISCO	8-11

Di periferiche come il drive o il modem ci occuperemo in un altro articolo. Nel frattempo ricordate di fare molta pratica: se disponete della stampante potrete divertirvi a stampare usando i vari set di caratteri che la macchina vi mette a disposizione, selezionabili inviando sul canale della stampante l'opportuno codice di controllo che potrete trovare sul manuale.
Anche questa volta terminiamo qui la nostra breve lezione sul BASIC del COMMODORE 64; per altre interessanti informazioni non dimenticate di leggere il prossimo numero.

Massimo Cellini

ISTRUZIONI PER LA CASSETTA DI PLAY ON TAPE COMPUTER N. 7

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)

ANAGRAMMI

Questo è un gioco divertente per trascorrere piacevolmente un po' di tempo da soli o in compagnia: potrai infatti organizzare delle vere e proprie gare, basate sulla bravura e sulla velocità. Il tuo VIC 20 ti proporrà delle parole anagrammate che tu dovrai cercare di decifrare in un tempo massimo di 30 secondi, prima che sia il computer stesso a darti la risposta giusta. Quando avrai scoperto di che parola si tratta, dovrai scriverla usando i tasti di controllo del cursore per spostarti lungo le lettere e, una volta fermo su quella che hai scelto, premi "space" per comunicarlo al VIC.

Se durante la composizione commetti un errore, premi il tasto DEL; se rinunci alla mano e vuoi conoscere subito la soluzione esatta, premi il tasto F1.

Per ogni quiz hai a disposizione 3 tentativi, superati i quali ti verrà conteggiato un errore; ricorda che anche una rinuncia è un errore. Dopo 5 sbagli consecutivi il gioco termina. Per ricominciare premi un tasto, altrimenti premi N e il computer ti darà il punteggio definitivo. Buon divertimento.

PONTICELLO

Inserito il nostro gioco leggete attentamente le istruzioni, potrete usare il joystick o le lettere E R T D G C V B con lo schema riprodotto sul vostro video e che potrete trascrivere per ricordare meglio. A questo punto scegliete il livello di difficoltà (1/5); all'inizio non sentitevi troppo sicuri, perché non è così facile come sembra.

Vi renderete conto che quell'omino dispettoso non esce mai dalla parte che voi vi aspettate e che, se non sarete più che abili e veloci, non riuscirete ad impedire al vostro protetto di precipitare nelle fredde acque del fiume che passa sotto i ponti che voi dovrete costruire rapidi come razzi. Attenzione perché i ponticelli non rimangono integri per troppo tempo e in realtà non avrete neanche un solo attimo di respiro. Forza: degli ingegneri abili come voi non devono certo lasciarsi impressionare dalle difficoltà!

PUZZLE

PUZZLE è un passatempo divertente ed utile per allenare la tua memoria visiva. La figura di partenza è quella di un luminoso paesaggio primaverile: cielo azzurro, sole, un albero carico di foglie e una strada di campagna che si perde in lontananza. In seguito lo stesso paesaggio ti verrà mostrato suddiviso in 9 tasselli disposti casualmente, a volte persino sottosopra: dovrai riordinarli per ricostruire la figura di partenza.

Per invertire la posizione di due tasselli premi successivamente i due numeri che li indicano; per raddrizzare un tassello storto, premi il suo numero più un tasto qualsiasi.

SONAR CAVE

Sei un esploratore che, per uno sfortunato contrattempo, si è perduto nelle viscere della terra. Per riconquistare la perduta libertà dovrai procedere a tentoni, caverna dopo caverna, servendoti soltanto di un sonar, la cui nota diventa più acuta via via che ti avvicini ad una delle uscite celate nella roccia. Mentre procedi nella tua ricerca, dovrai fare molta attenzione ai pericolosissimi pipistrelli che si lanceranno su di te per ucciderti con il loro morso velenoso: per difenderti hai a disposizione 3 mine che dovrai fare esplodere usando il tasto ←.

Per muoverti nel buio usa i tasti:

ϱ per camminare verso l'alto,
/ per dirigerti in basso,
= a destra
: a sinistra.

Il punteggio sarà calcolato in base al tempo impiegato a trovare un'uscita (hai a disposizione un massimo di 30 secondi), al quadro raggiunto e al numero di mine rimaste inesplose. Ad ogni porta scoperta, ti verranno concesse nuove possibilità di salvezza.

A questo punto non possiamo fare altro che augurarti...buona fortuna, esploratore!

JULIANUS DIES

In quest'epoca di curiosità e di precisione tecnologica non poteva mancare un programma come "JULIANUS DIES", che ci permetterà di conoscere, con rapidità e assoluta certezza, le fasi lunari secondo il calendario "Giuliano", in uso fino al 1581 e successivamente sostituito da quello attuale: potremo così scoprire (cosa molto utile per un astrologo o per uno studioso), quale giorno della settimana era, per esempio, il 5 maggio nel 320 a.C., senza incorrere in difficili calcoli e probabili errori grossolani.

Dunque, inseriamo "JULIANUS DIES"; apparirà la scritta: giorno, mese, anno.

Inseriamo allora la data che ci interessa conoscere, per esempio il giorno: 5,5,1956 e subito vedremo comparire sullo schermo le informazioni richieste:
anno giuliano, giorno, fase lunare.

Seguendo le istruzioni del computer otterremo risposte ancora più specifiche.

GIORNI FRA DUE DATE

All'inizio del programma "JULIANUS DIES" comparirà anche la scritta "GIORNI FRA DUE DATE", che ci darà l'opportunità di conoscere con estrema esattezza il numero dei giorni che intercorrono tra una data e un'altra.

Sul video apparirà la richiesta della prima data:
giorno, mese, anno;
scriveremo per esempio: 28,8,1953 e poi inseriremo anche la seconda data che ci interessa. Immediatamente il computer calcolerà l'esatto numero dei giorni trascorsi in tutti questi anni. Non sentitevi troppo vecchi quando scoprirete di avere più di 7000 giorni: non sono poi così tanti!

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

HAT IN THE RING

HAT IN THE RING è un gioco di simulazione di una campagna elettorale di due candidati (democratico e repubblicano) alla Presidenza degli Stati Uniti. Se uno dei tuoi sogni segreti è quello di vivere, anche soltanto per un giorno, l'affascinante e stimolante ruolo di leader politico, corri a provare questa nuova video-cassetta e armati di buon senso e di attenzione.

I giocatori sono due: scegli quale candidato vuoi essere e inserisci il tuo nome e quello del tuo avversario.

I due candidati avranno la possibilità, al loro turno, di prendere visione di una lista di operazioni da effettuare durante la propria campagna elettorale: potranno controllare i fondi a loro disposizione, essere informati sulla situazione politica dei singoli Stati e conoscere i risultati dei sondaggi di opinione per scegliere così la strategia più opportuna per aumentare i consensi. Per ottenere le informazioni richieste battere il tasto (da 1 a 7) corrispondente alla voce che vi interessa e subito dopo il RETURN.

La saggia amministrazione dei fondi destinati alla campagna elettorale potrà essere l'asso nella manica di ciascun candidato, quindi ricordatevi di tenere sempre d'occhio la situazione economica. Stanziare dei fondi potrà significare, nei singoli Stati, modificare a proprio favore le situazioni...ma attenzione a non andare fuori budget: è inconcepibile che un candidato alla Presidenza incapace di amministrare i propri fondi possa essere eletto!

Il gioco si articola su un totale di 20 turni durante i quali ogni candidato dovrà far valere le sue doti di astuzia e di saggia amministrazione al fine di trionfare nelle elezioni. Auguri!

DIETA

È ormai risaputo che mantenere il proprio corpo in condizioni di peso ideali, aiuta un individuo a sentirsi in forma sia fisicamente che psicologicamente, eliminando gran parte degli squilibri che la vita odierna, purtroppo, alimenta facilmente.

È chiaramente impossibile definire una dieta ideale valida per tutti: infatti, per esempio, una persona che soffre di diabete dovrà evitare gli zuccheri mentre un cardiopatico non dovrà consumare i grassi animali; ma un individuo in normali condizioni di salute potrà facilmente organizzare il proprio regime alimentare senza un diretto controllo medico, semplicemente razionando il numero di calorie da assumere nell'arco di una giornata.

Per stabilire una dieta corretta, bisogna tenere conto di alcuni fattori individuali, quali l'età, il tipo di struttura fisica, il lavoro fisico e mentale svolto.

Caricato il programma, il computer vi chiederà dunque alcune informazioni riguardo al vostro sesso, all'età, al peso, all'altezza e al vostro biotipo. Una volta forniti i vostri dati, conoscerete il peso ideale da raggiungere, anche in base al tipo di attività svolta.

Potrete decidere di organizzare la vostra dieta nel periodo di tempo proposto dal computer (90, 180 o 360 giorni), oppure scegliendo un numero di giorni minore o maggiore a seconda delle vostre esigenze personali: in questo modo vi sarà fornita la quantità di calorie da assumere quotidianamente.

Ricordatevi comunque che il programma è valido per persone che abbiano superato il dodicesimo anno di età, in quanto l'alimentazione dei bambini deve necessariamente essere sottoposta al controllo medico, per evitare errori che potrebbero rivelarsi dannosi proprio nel periodo della crescita.

Per conoscere il valore calorico dei singoli alimenti, vi rimandiamo al programma CALORIE, presente in questo stesso numero, di fondamentale importanza per organizzare con indicazioni sicure la vostra DIETA PERSONALIZZATA.

CALORIE

Abbinato a DIETA, presente in questo stesso numero, CALORIE è di fondamentale importanza per chi desidera seguire un corretto programma di alimentazione in base alle necessità personali e al proprio consumo calorico giornaliero, per riuscire finalmente a perdere i chili di peso superflui.

Il menù iniziale fornisce 4 voci:

1) lista per la scelta
2) totale calorie
3) lista alimenti scelti
4) fine programma

Battendo il tasto 1 otterrete la lista che contiene un gran numero di alimenti tra quelli consumati con maggiore frequenza:

inserendo il nome e la quantità scelta (in gr.) potrete conoscere il potere calorico del cibo che vi interessa.

Richiedendo la voce 2) totale calorie, scoprirete se il vostro pasto giornaliero rientra nei limiti calorïci imposti dalla dieta personalizzata.

In questo modo non rischierete più di "morire di fame" senza, alla fine, ottenere nessun risultato: per dimagrire non serve fare inutili sacrifici, ma basta seguire un metodo equilibrato di alimentazione che tenga conto delle esigenze personali.

POKER

Con il tuo C16 il poker diventa un "vizio" accessibile anche a te! Sfida subito il tuo computer in una divertente ed appassionante partita.

Entrambi avete a disposizione un capitale iniziale di L. 1.000.000: ricorda che ogni puntata non deve superare L. 200.000 e che ogni mano di poker costa L. 10.000 che verranno aggiunte o detratte, alla fine di ogni partita, dalla somma a tua disposizione.

Sul video appariranno le tue 5 carte: studiale bene e, se decidi di cambiarne alcune, batti i numeri corrispondenti e il tasto C: le carte scelte verranno sostituite da altre nuove. Alla domanda "Quanto punti?" batti la somma scelta e il RETURN. Subito dopo il computer ti darà l'indicazione del punteggio raggiunto da entrambi e, sia che vinci sia che perdi, la somma verrà automaticamente aggiunta o detratta dal tuo capitale o da quello del computer.

Batti il tasto G per iniziare una nuova partita o F se sei stanco di giocare.

15 3D

Non stupirti: è proprio lui, il vecchio gioco del quindici, il passatempo più conosciuto in tutto il mondo! Ma oggi è ancora più divertente perché...tridimensionale!

Le regole le conosci sicuramente: dovrai mettere nella giusta sequenza i numeri dall'1 al 15, sfruttando la casella vuota per poterli spostare. Usa i tasti:

ϱ per far scorrere il numero verso l'alto
/ per farlo scorrere verso il basso
* per muoverlo verso destra
: per portarlo a sinistra.

Il punteggio finale risulterà in base al numero delle mosse effettuate e al tempo impiegato per incasellare al loro posto i 15 numeri. Cerca di ottenere quindi un punteggio quanto più basso possibile; allenati più che puoi e sarai pronto a sfidare tutti i tuoi amici in questa gara di intelligenza e velocità!

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 7

VIC 20:

- ANAGRAMMI
- PONTICELLO
- PUZZLE
- SONAR CAVE
- JULIANUS DIES

C. 16 - PLUS/4:

- HAT IN THE RING
- DIETA
- CALORIE
- POKER
- QUINDICI 3D

EV EDITORIALE VIDEO