AmigaBasic: trattamento file

di Andrea de Prisco

Concludiamo con questo articolo la nostra carrellata sull'AmigaBasic. L'argomento che tratteremo questo mese riguarda la gestione dei file, sequenziali o ad accesso diretto, direttamente da Basic.

Sequenziali o random?

File in inglese vuol dire raccoglitore, cartellina. In *computerese*, il file è un insieme oggetti omogenei inseriti, in qualche modo, in una qualsiasi memoria centrale o di massa. La prima operazione da compiere per utilizzare un qualsiasi file, sarà la creazione del file stesso o, se questo è già stato creato precedentemente, la cosiddetta «apertura». Analogamente, dopo averlo utilizzato per le operazioni che illustreremo più avanti, dovremo «chiuderlo». Tanto l'operazione di creazione, apertura che quella di chiusura, sono possibili tramite determinati comandi del linguaggio di programmazione che stiamo adoperando. Al momento della creazione del file è necessario indicare per esso un nome al fine di poterlo nuovamente riferire quando ne avremo nuovamente bisogno. Oltre a questo, dal momento che generalmente serve di poter tenere aperti più file contemporaneamente, al momento dell'apertura di un file associamo ad esso un numero indicativo (volendo diverso di volta in volta) che utilizzeremo per riferire le operazioni che compiremo in seguito ad un determinato file invece che ad un altro.

I file si distinguono essenzialmente per il tipo di accesso che possiamo compiere. Il primo tipo di file «inventato» fu il file ad accesso sequenziale. Ciò esclusivamente per il fatto che le memorie di massa dei primi calcolatori utilizzavano nastri magnetici che, come avviene anche per le musicassette, possono essere letti solo sequenzialmente: se ad esempio la testina di registrazione si trova a 10 metri di nastro dall'inizio, per raggiungere il punto posto a 20 metri dall'inizio potremo solo scorrere (sequenzialmente) i dieci metri che ci separano dalla nostra meta e non saltare lì direttamente. Così i primi file avevano una struttura «obbligata» dal tipo di supporto sul quale dovevano giacere. Creato il file, il dispositivo cercava una zona vuota di nastro ed era pronto a ricevere, sequenzialmente, i dati, che il computer gli mandava. Analogamente, in lettura, riposizionata la testina sul punto di inizio del file, il dispositivo restituiva all'unità centrale i dati nello stesso ordine in cui gli erano pervenuti.

I file ad accesso sequenziale, come avrete notato, vanno molto bene quando si tratta di memorizzare «tutto d'un fiato» dati che riutilizzeremo in seguito nella loro intierezza, ad esempio quando dobbiamo salvare pezzi di memoria centrale da ripristinare in un secondo momento. Se però i nostri dati sono strutturati e siamo interessati ad accedere a singoli elementi casualmente, il file di tipo sequenziale non va più bene. Ma soprattutto non va più bene il dispositivo a nastri magnetici: occorre inventare un nuovo sistema di memorizzazione, di per sé, ad accesso diretto. Questo è quanto si sono detti gli ingegneri informatici di una trentina di anni fa: ...e fu il disco.

Che il disco magnetico permetta accessi diretti è cosa ovvia: pensate nuovamente all'analogo supporto musicale: se stiamo ascoltando il brano 3 e vogliamo saltare al brano 6 non dobbiamo far altro che afferrare il braccio e portarlo sul punto desiderato (senza dunque «passare» per i brani 4 e 5). I file ad accesso diretto (detti anche random) permettono dunque di memorizzare, e analogamente leggere o modificare, «singole» registrazioni.

La domanda da porsi è questa: se ormai le unità a nastro praticamente non esistono più, perché non parlare sempre e solo in termini di file ad accesso casuale? Risposta: perché... non è vero. Innanzitutto i nastri non sono scomparsi dalla circolazione (sono comunque un ottimo supporto per memorizzare, a basso costo, «roba» sequenziale, vedi ad esempio gli streamer per eseguire backup di hard disk) e poi, anche su disco, i file sequenziali sono ugualmente comodi per lo stesso motivo. Infatti, mentre per scaricare sequenzialmente qualcosa sul disco è sufficiente aprire il file, buttare dentro la roba e richiuderlo, coi file ad accesso diretto le cose si complicano un po' sia per l'utente che per il sistema... ergo, se non siamo interessati all'accesso diretto, meglio semplificare le cose.

File e AmigaBasic

Dopo questa lunga carrellata storico-informatica sui file in generale, entriamo nell'argomento specifico che, qualora l'avessimo dimenticato, riguarda il trattamento file da AmigaBasic. In questo linguaggio di programmazione, come era prevedibile, è possibile creare e quindi utilizzare ambedue i tipi di file sopra descritti. Per i file sequenziali, inoltre è possibile «appendere» ad un file già esistente nuovi dati senza perdere quelli precedentemente inseriti.

Per creare o aprire un file sequenziale occorre indicare il nome del file,

Figura 1

```
OPEN "pippo" FOR OUTPUT AS #1
  FOR i=1 TO 100
     WRITE #1,i
  NEXT
CLOSE #1
```

Figura 2

```
OPEN "pippo" FOR INPUT AS #1
  WHILE NOT EOF(1)
     INPUT #1,i
     PRINT i
  WEND
CLOSE #1
```

Figura 3

```
OPEN "R",#1,"pippo",4
FIELD #1, 4 AS N$
  FOR i =1 TO 100
     RSET N$ = MKS$(i )
     PUT #1,i
  NEXT
CLOSE #1
```

Figura 4

```
OPEN "R",#1,"pippo",4
FIELD #1, 4 AS N$
  FOR i =1 TO 100
     GET #1,i
     PRINT CVS(N$)
  NEXT
CLOSE #1
```

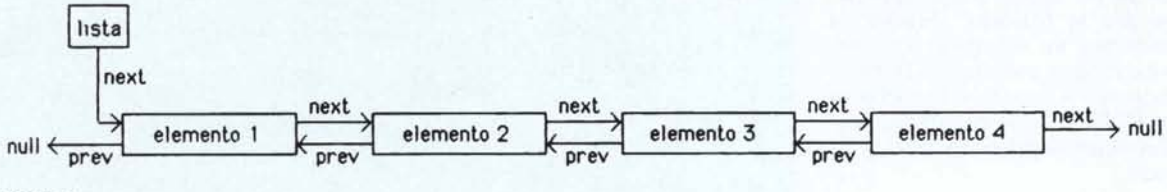
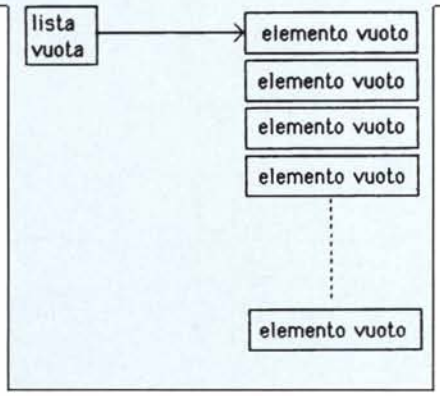


Figura 5



un numero indicativo (come detto prima), e il tipo di operazione che ci accingiamo a compiere: lettura, scrittura o «append». Esistono due sintassi per raggiungere tale scopo, una più prolissa, l'altra più sintetica. Immaginiamo ad esempio di creare un nuovo file di nome «Pippo» e decidiamo di assegnare ad esso il numero indicativo 1. Essendo un caso di creazione, l'unica operazione che potremo compiere su di esso sarà la scrittura. Con la prima sintassi scriveremo:

```
OPEN "Pippo" FOR OUTPUT AS #1
```

che tradotto in italiano vuol dire proprio «apri Pippo per output come 1». Con la seconda sintassi scriveremo:

```
OPEN "O", #1, "Pippo"
```

che ovviamente è intraducibile...

Per «stampare» qualcosa nel nostro file è possibile procedere in due distinti modi: o con dei normalissimi PRINT reindirizzati verso il file (nel nostro caso useremmo quindi PRINT #1) oppure un più appropriato comando WRITE che ha il grosso vantaggio di non sprecare spazio sul disco: una serie di dati «spediti» via WRITE occupa esattamente lo spazio necessario, ovviamente separando i vari dati da un LINE FEED. In figura 1 è mostrato un programmino che apre un file, stampa al suo interno i primi 100 numeri.

In figura 2 è mostrato il corrispondente programma in grado di leggere i dati appena inseriti e stamparli sul video. La funzione EOF(1) restituisce il valore «vero» solo quando è stato letto l'ultimo dato dal file: l'1 (uno) racchiuso tra parentesi è l'indicativo del file in questione. Il comando che effettua la lettura vera e propria è INPUT# seguito dall'indicativo e dalla o dalle variabili cui assegnare il dato in arrivo dal disco. Da notare che con i file sequenziali non siamo legati a tipi di dato in quanto possiamo «buttare» dentro, alla rinfusa, stringhe e numeri come ci pare. Ovviamente, al momento di rileggere il file, dovremo ricordarci i tipi di dato via via in arrivo, per non incorrere in messaggi d'errore «Type mismatch». Infatti le stringhe vengono salvate racchiuse tra apici e quindi non c'è modo di prelevare con una variabile numerica ciò che numerico non è, come ad esempio la stringa «123».

Per appendere altri dati ad un file sequenziale già esistente indicheremo la parola chiave APPEND nel comando di OPEN al posto dei già visti INPUT e OUTPUT. Nel caso in cui vogliamo utilizzare la sintassi più breve, indicheremo tra apici la lettera "A". Dunque:

```
OPEN "Pippo" FOR APPEND AS #1
```

oppure

```
OPEN "A", #1, "Pippo"
```

Infine, per conoscere la dimensione di un file sequenziale, dopo averlo aperto (FOR INPUT) useremo la funzione LOF() indicando, come il consueto, l'indicativo dei file tra le due parentesi.

File Random

In AmigaBasic i file random, o ad accesso casuale che dir si voglia, sono composti da elementi di tipo record. Nella accezione più classica un dato di tipo record è un insieme di oggetti disomogenei, ognugno etichettato da un nome per poterlo riferire, associati ad un nome globale che indirizza l'intera struttura. Un esempio di record potrebbe ad esempio essere (stiamo parlando in generale, non di AmigaBasic) la variabile Persona, composta da vari campi come Nome, Cognome, Età. Persona identifica dunque una struttura formata da due oggetti di tipo stringa etichettati da «Nome» e «Cognome», e da un oggetto di tipo intero etichettato dal nome «Età». Per associare dei valori alla variabile persona dovremo dunque eseguire tre assegnamenti distinti:

```
Persona.Nome = "Andrea"
Persona.Cognome = "De Prisco"
Persona.Età = 26
```

In AmigaBasic il tipo di dato record, purtroppo, non esiste. Come detto prima però le singole registrazioni di un file di tipo random hanno una struttura di tipo record anche se, stranamente, i singoli campi possono essere solo di tipo stringa. Al momento della creazione di un file random, oltre all'indicazione del nome e dell'indicativo come visto prima per i file sequenziali, occorre definire alcuni parametri di fondamentale importanza. Innanzitutto occorre indicare la lunghezza massima, espressa in caratteri, di ogni registrazione. Subito dopo nomi e dimensioni dei vari campi di cui ogni registrazione sarà composta. Ovviamente i «vari» campi possono essere anche uno solo nel qual caso si procederà come nell'esempio di figura 3 e 4. Commentiamolo brevemente.

La prima istruzione di figura 3:

```
OPEN "R", #1, "Pippo", 4
```

indica che stiamo aprendo un file "R"andom, al quale associamo l'indicativo 1, il nome del file è "Pippo" e la lunghezza di ogni registrazione sarà pari a 4 caratteri. Con la linea successiva:

```
FIELD #1, 4 AS N$
```

indichiamo che le nostre registrazioni saranno formate da un solo campo, di nome N$, lungo ovviamente 4 caratteri.

Per assegnare ai campi del record dei valori, si utilizza il comando di assegnamento RSET o LSET a seconda se desideriamo un allineamento a destra o a sinistra del valore nel campo. Ovviamente ciò è valido solo se le dimensioni del campo sono maggiori

del valore assegnato, altrimenti allineare a destra o a sinistra è, banalmente, la stessa cosa. Nella fattispecie, dato che la funzione MKSS, la quale trasforma un «singola precisione» in una stringa codificata, restituisce esattamente 4 caratteri, la scelta di utilizzare RSET è stata puramente casuale. Tornando al programmino di figura 3, con:

```
RSET N$ = MKS$(I)
```

settiamo il campo N$ col «codificato» di I e con:

```
PUT #1,I
```

diamo ordine di registrare nella i-esima posizione il contenuto di N$ precedentemente assegnato. I più attenti avranno capito che tale programmino non fa altro che scrivere nelle prime cento posizioni del file random «Pippo» i primi cento numeri: un po' come faceva il programma di figura 1, solo che adesso utilizziamo un file di tipo diverso.

In figura 4 abbiamo messo il programmino che, come quello di figura 2, ripesca uno per uno i numerini e li stampa sullo schermo. Le operazioni di apertura file sono le medesime di prima (con i file random non si distingue tra apertura per inserimento, lettura o aggiornamento record) e le differenze riguardano il comando GET che serve per «pescare» il record desiderato. Eseguita infatti questa operazione le variabili definite col comando FIELD sono aggiornate e possiamo utilizzarle come vogliamo. Da notare la funzione CVS, corrispondente alla MKS$ di cui sopra, che serve per decodificare la stringa registrata e ottenere nuovamente il valore numerico. Sul manuale dell'AmigaBasic troverete le altre funzioni che convertono dati numerici differenti, come i doppia precisione o gli interi lunghi.

L'esempio

Il programma listato in queste pagine è una banale applicazione di quanto detto fin ora riguardo i file random. E un piccolo indirizzario con chiave unica sul Cognome i cui elementi sono collegati a lista doppia ed ogni inserimento mantiene le registrazioni sempre in ordine alfabetico. Lo schema della lista è mostrato in figura 5. I puntatori «di ritorno» ovvero quelli etichettati «prev» pur essendo costantemente aggiornati dal programma non vengono utilizzati. Il programma infatti è particolarmente «basico» dovendo servire solo come esempio finale dell'articolo, ma è già pronto per essere modificato a piacimento da chi

```
MENU 1,0,1,"Indirizzi    "
MENU 1,1,1,"Creazione    "
MENU 1,2,1,"Aggiornamento"
MENU 1,3,1,"Consultazione"
MENU 1,4,1,"Inserimento  "
MENU 1,5,1,"Fine lavoro  "

MENU 2,0,0,""
MENU 3,0,0,""
MENU 4,0,0,""

ON MENU GOSUB scelta
MENU ON
CLS
GOSUB stampa

WHILE 1
WEND

stampa:
PRINT "scegli dal menu a discesa..."
RETURN

scelta:
ON MENU(1) GOTO crea,agg,con,ins,fine

crea:
  OPEN "R",#1,"indirizzi",83
  FIELD #1, 12 AS COGNOME$, 13 AS NOME$, 35 AS RECAPITO$, 15 AS TELEFONO$, 4 AS nex$, 4 AS prev$
  LSET nex$ = MKS$(0)
  LSET prev$ = MKS$(2)
  PUT #1,1
  CLOSE #1
  GOSUB stampa
RETURN

ins:
  CLS
  OPEN "R",#1,"indirizzi",83
  FIELD #1, 12 AS COGNOME$, 13 AS NOME$, 35 AS RECAPITO$, 15 AS TELEFONO$, 4 AS nex$, 4 AS prev$
  WHILE 1
  c$=""
  INPUT "Cognome  ";c$
  IF c$="" THEN CLOSE #1:GOSUB stampa:RETURN
  INPUT "Nome     ";n$
  INPUT "Recapito ";r$
  INPUT "Telefono ";t$
  GET #1,1
  la=CVS(prev$)
  t=1
  WHILE t
    IF CVS(nex$)=0 THEN
       LSET COGNOME$=c$
       LSET NOME$=n$
       LSET RECAPITO$=r$
       LSET TELEFONO$=t$
       LSET nex$=MKS$(0)
       LSET prev$=MKS$(LOC(1))
       PUT #1,la
       t=0
    ELSE
       GET #1, CVS(nex$)
       IF c$<COGNOME$ THEN
          LSET COGNOME$=c$
          LSET NOME$=n$
          LSET RECAPITO$=r$
          LSET TELEFONO$=t$
          LSET nex$=MKS$(LOC(1))
          PUT #1,la
          t=0
       END IF
    END IF
  WEND

     n=CVS(nex$)
     p=CVS(prev$)
     i=LOC(1)
     IF n <> 0 THEN
        GET#1,n
        LSET prev$ = MKS$(i)
        PUT#1,n
     END IF
     GET#1,p
     LSET nex$ = MKS$(i)
     PUT#1,p
     GET #1,1
     LSET prev$=MKS$(i+1)
     PUT #1,1
  WEND


con:
  OPEN "R",#1,"indirizzi",83
  FIELD #1, 12 AS COGNOME$, 13 AS NOME$, 35 AS RECAPITO$, 15 AS TELEFONO$, 4 AS nex$, 4 AS prev$
```

```
  CLS
  WHILE 1
  c$=""
  INPUT "Cognome (@=tutti)";c$
  IF c$="" THEN CLOSE #1 :GOSUB stampa: RETURN
  GET#1,1
  WHILE CVS(nex$)<> 0
    GET#1,CVS(nex$)
    IF LEFT$(c$+"            ",12)=COGNOME$ OR c$="@" THEN
       PRINT COGNOME$;NOME$;
       PRINT RECAPITO$;TELEFONO$
    END IF
  WEND
  WEND

agg:
  OPEN "R",#1,"indirizzi",83
  FIELD #1, 12 AS COGNOME$, 13 AS NOME$, 35 AS RECAPITO$, 15 AS TELEFONO$, 4 AS nex$, 4 AS prev$
  WHILE 1
  c$=""
  INPUT "Cognome ";c$
  IF c$="" THEN CLOSE #1 :GOSUB stampa: RETURN
  GET#1,1
  WHILE CVS(nex$)<> 0
    a=CVS(nex$)
    GET #1,a
    IF LEFT$(c$+"            ",12)=COGNOME$ THEN
       n$="":PRINT "Nome     ";NOME$;INPUT n$;IF n$<>"" THEN LSET NOME$=n$
       r$="":PRINT."Recapito ";RECAPITO$;INPUT r$;IF r$<>"" THEN LSET RECAPITO$=r$
       t$="":PRINT "Telefono ";TELEFONO$;INPUT t$;IF t$<>"" THEN LSET TELEFONO$=t$
       PUT #1,a
    END IF
  WEND
  WEND

fine:
  MENU RESET
END
```

ne avrà voglia. Coi doppi puntatori, infatti, potremo scorrere facilmente la lista nei due versi, avendo la certezza di manipolare sempre oggetti in ordine alfabetico (per Cognome).

I campi delle registrazioni sono Nome, Recapito e Telefono per quanto riguarda la parte visibile all'utente finale, più due campi di servizo Nex e Prev che contengono rispettivamente le posizioni del prossimo e del precedente elemento nell'ordine alfabetico. Il primo record del file random non è utilizzato per gli indirizzi, ma anch'esso per servizio: i campi Nome, Recapito e Telefono sono costantemente vuoti, il campo Nex contiene l'indirizzo del primo elemento (in ordine alfabetico) in lista, il campo Prev contiene la posizione del primo elemento vuoto, che sarà utilizzato al prossimo inserimento.

Ah, dimenticavo! Qualora vi sfuggano concetti come lista, puntatori, chiavi o altro, vi rimando al numero 50 di MCmicrocomputer dove, a pagina 116, nella rubrica Appunti di Informatica, non abbiamo parlato d'altro. Buona lettura.

MC

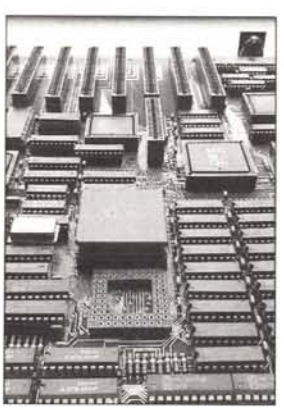
SALVA IL TUO PC DA UN PENSIONAMENTO PRECOCE.

ORDINA OGGI LA TUA NUOVA MOTHERBOARD BRAINSTORM 386 ST

80386 ST/MB - LIT. 3.800.000

Dai al tuo PC nuova linfa vitale! Con la nostra motherboard 386 il PC, PXC/XT o compatibile sarà un degno rivale in velocità dei migliori sistemi 386 in circolazione. Addirittura più veloce. Questo per il Megabyte di RAM ad alta velocità e lo zoccolo per il coprocessore 80387 per velocità esplosive mai raggiunte sinora. Per lasciare il pensionamento fuori della porta 80386 ST/MB è compatibile con il PC/AT (BIOS e I/O) e vi permette di usare la nuova generazione di DOS, l'0S/2 e il PC/MOS 386.
Abbiamo anche incluso due slot di espansione a 16 bit per le più recenti schede di espansione.
Nessuna scheda acceleratrice potrebbe darti tanta versalità.

Hauppauge!



Con la potenza del 386 e vera compatibilità software AT, il tuo lavoro, il Desktop Publishing e le tue applicazioni ingegneristiche avranno una sferzata di nuova produttività.

Specifiche Tecniche:

- 16 Mhz 80386 - 1 Megabyte di inteleaved RAM a 100 nsec - I/O e BIOS compatibile AT per il supporto dell'0S/2 - sette slot espansione a 8 bit - due slot espansione a 16 bit - uno slot di espansione RAM a 32 bit (max 12 megabyte) - coprocessore matematico 80387 opzionale.

Per maggiori dettagli e informazioni chiamaci oggi!

Importato e distribuito in Italia da:

gesin trade

GESIN TRADE srl - Via Virginio Orsini, 19 - 00192 Roma - Tel. 06/385177/381950/3595856