di Raffaello De Masi

Le operazioni di I/O

terza parte

Per un linguaggio così particolare e finalizzato come il Prolog, l'uso delle periferiche e delle tecniche di I/O è tanto importante che, a buon motivo, è possibile dire che una corretta manipolazione delle periferiche e dei file risolve più del 50% delle difficoltà nella redazione di un programma.
Ciò detto, continuiamo il nostro discorso sulle tecniche di I/O dando un'occhiata ai file ad accesso casuale, l'altra grande famiglia dei file su disco. Ma prima di arrivare a ciò occorre ricordare un piccolo particolare, di grande utilità, che mi è sfuggito la volta scorsa

Alcuni predicati speciali, specifici di Turbo Prolog

Esistono alcuni predicati, particolari di questo linguaggio, che non hanno corrispettivo in altri idiomi, e che risultano particolarmente utili in certe occasioni particolari; uno di essi è [file_str].

Questo predicato permette di leggere una stringa di caratteri da un file DOS e di conservarlo in una variabile di stringa all'uopo dichiarata, senza dichiarare il file e assegnare ad esso il nome simbolico. Si tratta di un operatore certo non molto efficiente e che non conviene usare in maniera generalizzata, ma può essere utile, ad esempio, se si desidera eseguire un controllo su un file o cercare una particolare stringa smarrita negli oscuri meandri dei file. Il predicato, in maniera piuttosto bovina, continuerà a leggere caratteri uno alla volta dal file esaminato fino a che si verifica almeno una delle seguenti condizioni: l'incontro di un marker [End_Of_File] (rappresentato generalmente da un simbolo [Ctrl-Z]) oppure la lettura e l'immagazzinamento nella variabile di 64 Kb di informazioni. Ripetiamo che non è necessario, in ogni caso, aprire il file o definirlo come controparte delle operazioni di I/O per poter eseguire questa operazione.

L'affine, tanto per usare un termine legale, di [file_str] è il predicato [consult]; più raffinatamente del precedente, esso esige che almeno il nome DOS del file sia definito, anche se non è necessario utilizzare un nome di file simbolico. Come [file_str] non richiede apertura o riassegnamento dell'indirizzo di lettura. Al contrario del predicato precedente, i dati letti con [consult] non vengono immagazzinati nella stringa ma direttamente immessi nella base di dati aperta dal programma. Generalmente, [consult] viene utilizzato per leggere dati in una base salvata col predicato [save]. L'operazione tipica è rappresentata dalla creazione della base di dati in editor, il salvataggio di questa in un file, e la consultazione dello stesso file da parte di un programma che usa i dati conservati nel file stesso.

Una notevole facilitazione di questo predicato rispetto a quelli standard di lettura è rappresentato dal fatto che esso legge anche archivi costruiti con word processor (invece che con l'editor proprio di Turbo Prolog). Non conviene però abusare più di tanto di questa tecnica, visto che un qualsiasi errore o disfunzione di lettura determina un errore di lettura e questa non va più avanti (a meno di complesse procedure di maneggio di errore; ma in questo caso il vantaggio dell'uso del predicato è meno vanificato dall'adozione di tecniche di programmazione aggiuntive non certo semplici).

File ad accesso casuale

Tutti i predicati finora definiti servono per leggere, scrivere o comunque maneggiare informazioni ordinate nei file sequenzialmente (vale a dire come una lunga stringa di dati, come le chiama Corrado, degli ASCII chilometrici). Anche se questi dati includono dei ritorni di riga o dei cambi-pagina, che li fanno somigliare a collezioni di informazioni diverse di registrazioni o informazioni, dal punto di vista di Prolog non c'è niente di diverso da una lunghissima stringa di caratteri alfanumerici. È il programma (e non il file) che provvede a leggere virgole, TAB, CR, LF o altri separatori per trasformare questa lunghissima stringa in una serie di informazioni maneggiabili dalla base di dati.

Potrebbe però darsi il caso che si debbano individuare dati in maniera mirata, vale a dire leggere record nel file senza per questo scorrerlo tutto, né caricarlo interamente nel sistema. Prolog fornisce una tecnica di accesso casuale ai file contenenti record o informazioni, attraverso una oculata combinazione dei predicati [openmodify] e [filepos].

Il primo predicato, [openmodify] ci è già noto; il suo uso primario nella manipolazione dei file random è legato alla tecnica di scansione dei record presenti nel file, alla localizzazione e lettura dell'informazione desiderata, alla scrittura eventuale di nuovi dati sul disco, cosa che [openmodify] riesce ad eseguire nel migliore dei modi.

Il predicato [filepos] tiene traccia di dove il puntatore è posizionato nel file, sia per operazioni di lettura che di scrittura, permettendo tra l'altro di cambiare questa locazione; esso maneggia tre argomenti:

- il nome di file simbolico maneggiato
- un intero che rappresenta la posizione del file
- un secondo intero, con valore limitato a 0, 1 o 2, che definisce il punto dal quale la posizione del file va misurata.

Tutti i file di programma conservano le informazioni sotto forma di byte. Un byte è, come noto, equivalente a una singola lettera o numero. Il valore relativo alla posizione nel secondo argomento indica di quanti byte è sfalsata la lettura (byte definiti secondo la tabella successiva), secondo quanto definito dal terzo argomento. Vale a dire che da questo punto in poi il sistema provvederà a leggere o scrivere i dati. Da una attenta analisi delle funzioni di modo, definite nella tabella a) si vedrà che nella maggior parte dei casi si utilizzerà il modo 1, poche volte il modo 2 e probabilmente mai (non si capisce poi bene a che cosa possa servire) il valore 3.

E facciamo un esempio; immaginiamo di avere un file di dati random, con ogni record rappresentato da un blocco di 200 byte. Potrebbe essere, per ipotesi, una raccolta di titoli di libri o dischi. Trattandosi di un file random, ogni record ha una lunghezza fissa (anche se non è pieno); in questo modo ogni record è direttamente locabile e prevedibile nella sua posizione fisica. Il primo byte è alla posizione 0, il secondo è alla posizione 200, il terzo alla 400 e così via.

L'algoritmo di sviluppo di un piccolo programma in Turbo Prolog per la lettura (e l'eventuale aggiornamento) del file è esposto in figura. Esso è rappresentato da una serie di passaggi piuttosto intuitivi, che si traducono effettivamente in altrettante righe di programma.

Tradotto in programma, la sequenza diviene quella della figura b; ovviamente è possibile rendere più elastico il tutto cambiando il valore fisso di 128 con una variabile cui assegnare, in input, la lunghezza del file; una volta eseguite le operazioni di ricerca del record è sempre possibile, con la grande elasticità fornita dai predicati specifici di manipolazione delle stringhe, estrarre porzioni del record da modificare, eseguire eventuali modifiche e riconservare il tutto senza eccessive difficoltà.

Usando il predicato [filepos], può essere interessante o utile sapere dove fisicamente finisce il file, in modo da non tentare di leggere oltre; a tale compito assolve il predicato [eof], l'end of file ben noto anche in altri linguaggi, che abbinato al nome simbolico del file destinato all'esame da valore [true] o [false] se il [filepos] è posizionato o meno alla fine del file.

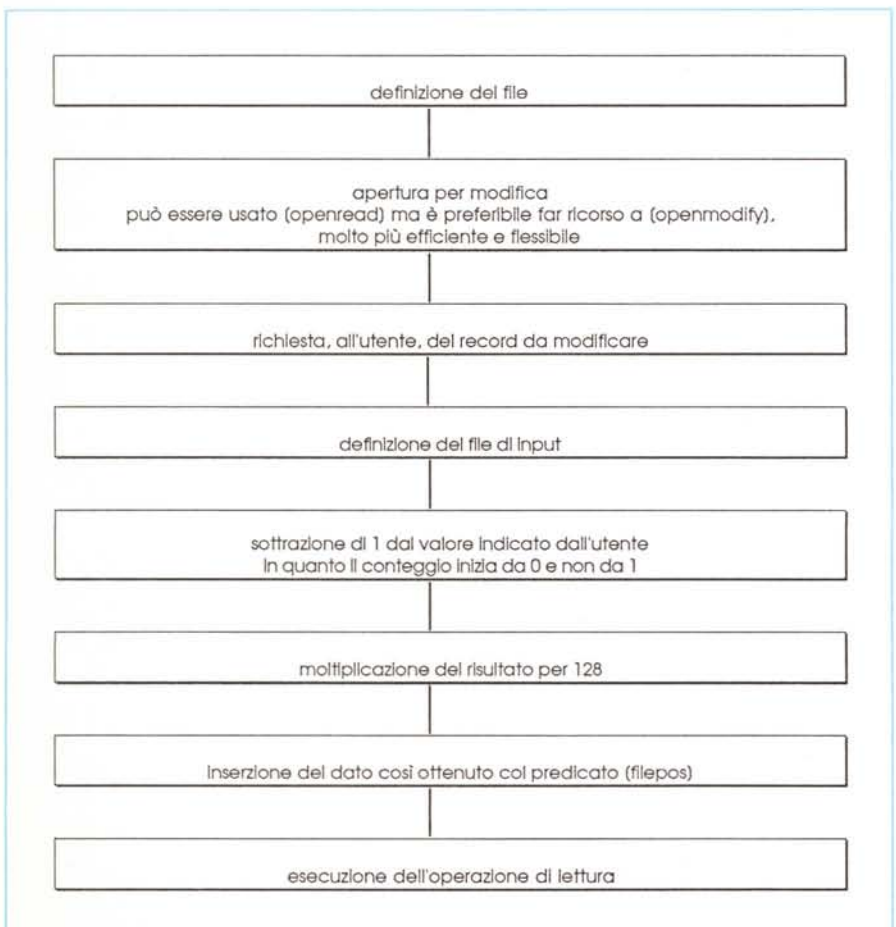

Figura a - Algoritmo d'uso del predicato (filepos) nell'aggiornamento di un file.

Comandi finalizzati al DOS

Oltre ai comandi finora descritti, specifici dell'ambiente Prolog, Turbo Prolog possiede, per il particolare ambiente per cui è sviluppato, una serie di statement, di predicati che consentono di accedere direttamente ai più utili e comuni comandi del sistema operativo MS-DOS, direttamente dall'interno del linguaggio. In quest'ultimo scorcio di articolo vedremo come, attraverso Prolog, è possibile accedere direttamente a comandi DOS, in particolare per quanto attiene alla manipolazione dei file.

Alcune applicazioni particolarmente complesse (come ad esempio un word processor, o anche un più semplice programma che utilizza basi di dati particolarmente estese) richiedono talvolta la creazione di file temporanei che saranno poi cancellati alla fine dell'uso dell'applicazione, senza che per questo l'utilizzatore sia chiamato ad eseguire operazioni di pulizia sulla memoria di massa. In altri termini la creazione e la successiva cancellazione di questi archivi temporanei deve essere del tutto trasparente per l'utilizzatore. La cosa può essere direttamente gestita dal programma-applicazione attraverso il predicato [deletefile].

Si tratta di un comando che è la copia carbone del comando DOS «ERASE» o «DEL»; esso manipola, come unico argomento, un nome DOS, fisicamente presente sul disco, e non un nome simbolico; il suo effetto, come prevedibile, è quello di cancellare il file nominato dalla memoria di massa corrente.

Altro predicato direttamente mutuato dal sistema operativo è [dir], che mostra la directory attiva della corrente memoria di massa. Esso funziona egualmente al suo omonimo [DIR] di DOS, e permette di visualizzare i file presenti che

```
openmodify(file_di_lavoro, "DISCHI").
write("Quale record desideri modificare ?").
readint(Numero_del_record).
filepos(file_di_lavoro, 128*(numero_del_record - 1),0.
readevice(file_di_lavoro).
readln(Nome_del_disco).
write(Nome_del_disco).
ecc.
```

Figura b L'algoritmo di figura A tradotto in programma.

possono essere selezionati per il successivo caricamento. Esso maneggia tre argomenti: il «pathname», della directory nella quale è custodito il file su cui si desidera lavorare, una stringa di individuazione del tipo di file, che individua l'eventuale suffisso determinante la «qualità» dei file da ricercare, e la variabile in cui il nome del file sarà custodito.

L'uso più comune di questo predicato è quello di leggere la lista dei file Prolog creati dall'applicazione corrente, per una eventuale scelta del soggetto da aprire.

Un esempio d'uso piuttosto corrente è:

```
dir("C:\","*.pro",Nome_del_file).
```

che legge la lista dei file di tipo «.pro» (quelli creati col Prolog sulla directory corrente del disco rigido).

Degno compagno di [dir] è l'altro predicato orientato al DOS, [disk], che mostra la corrente directory e consente il cambio della stessa e del drive corrente.

Se si usa un disco, con una variabile come suo argomento, il corrente drive e la pathname utilizzata sarà quella letta dal comando; un uso più sofisticato del comando può essere quello di abbinare i dati ricavati da [dir] con quelli ottenuti dal pathname per poterli utilizzare in una ricerca e modifica di archivi efficiente. Ma si tratta di un uso già più specializzato dei comandi, che va tagliato effettivamente per le esigenze del programma che si sta scrivendo, e non può essere perciò generalizzato.

E per concludere, parleremo del predicato [renamefile]; dall'uso piuttosto intuitivo, consente di cambiare il nome a un file DOS; esso maneggia due argomenti e il suo uso è del tutto simile al [rename] del DOS; il primo argomento è il nome di un file esistente, deputato alla rinominazione, il secondo è il nuovo nome che sostituirà il primo; anche in questo caso l'uso di un corretto e valido pathname permette di salire e scendere nelle directory a piacimento.

Anche stavolta abbiamo finito l'articolo esaurendo l'argomento relativo all'I/O su file; la prossima volta ci interesseremo ancora di I/O, ma per ciò che riguarda lo schermo e le finestre su di esso visualizzate; a risentirci.

MK