di Raffaello De Masi

Le operazioni di I/O

seconda parte

Resta, dopo quanto abbiamo detto la volta scorsa, ancora aperta l'esigenza di «comunicare» con l'esterno, in particolare con le periferiche. Prolog possiede, non meno di altri linguaggi, mezzi idonei a colloquiare con le periferiche ad esso collegate, e dispone di una serie di operatori per definire o dichiarare nuove periferiche e nuove vie di accesso ad esse. Questo porta automaticamente a discutere della tecnica d'uso dei file e delle modalità di modifica di questi. Leggendo quanto segue si vedrà come il nostro, pur non discostandosi particolarmente da tecniche già viste in altri idiomi, possiede quel tanto di originalità che gli consente di adattarsi specificamente alla particolare problematica che è chiamato a trattare

Le periferiche e la loro gestione

Nella sua allocuzione più comune, si intende come «periferica» un oggetto fisico collegato ad un altro che lo gestisce; tanto per fare un esempio, un campanello è una periferica gestita da un interruttore a pulsante; nella sua più ampia accezione una periferica non è solo rappresentata da oggetti comandati o regolati elettronicamente; anche una macchina, come, ad esempio la frizione di un'auto comandata da un pedale, è una periferica, ancorché sui generis.

Una periferica propria di un computer è generalmente individuabile come un oggetto fisico collegato alla macchina principale da esso comandato; una stampante, uno scanner, un'unità a dischi, un plotter, la stessa tastiera, sono periferiche; generalizzando potremo definire periferiche qualunque accessorio destinato a ricevere o trasmettere informazioni all'unità centrale; così, andando nei campi più avanzati delle applicazioni, saranno periferiche una penna ottica, ma anche un sistema di acquisizione acustico, un interprete di voce e un allarme automatico. Il linguaggio assume che la via di input standard sia la tastiera, a meno che, attraverso gli opportuni predicati, l'indirizzo di input sia cambiato usando predicati di cui parleremo tra poco. È ovviamente possibile stabilire «canali» per l'input, mantenendo contemporaneamente attivi sia la tastiera, sia altri indirizzi deputati alle operazioni di input stesso.

Analogamente, l'output standard è indirizzato allo schermo, che è considerato periferica di default. Ma, al contrario di quanto visto per le periferiche di input, Prolog può avere due altre periferiche di output, utilizzabili in alternativa, senza che ci sia la necessità di specificare la loro esistenza o la loro presenza fisica.

Ovviamente il programma in Prolog dovrà contenere le necessarie informazioni e istruzioni per indirizzare acconciamente l'output ad esse. Le due periferiche di cui si dice sono quelle collegate, in un PC, alla porta stampante e alla porta seriale; la prima viene indirizzata attraverso il nome stesso della stampante (in molti linguaggi si accede alla porta della stampante attraverso la denominazione LPT, o altri acronimi o nomi dedicati); la porta seriale, sempre nello stesso caso, è raggiunta attraverso la denominazione «COM1», anche se questa denominazione non è del tutto standard e può variare a seconda del costruttore della specifica macchina.

Ancora, i file su disco, che normalmente contengono fatti, e basi di dati possono essere usati come indirizzi di input e output; al contrario di altre tipologie di linguaggio viste altrove, non esiste un nome di file di default o standard: occorre specificatamente identificare il file su cui si desidera lavorare, secondo una tecnica che vedremo tra poco.

Per dichiarare nuovi indirizzi di I/O esistono due predicati dedicati; [readdevice] e [writedevice], destinati ambedue a cambiare gli indirizzi di input e output dalla configurazione standard Prolog ad altri indirizzi.

In ogni caso, l'argomento fornito al predicato è il nome dell'indirizzo dal quale il futuro input sarà ottenuto o al quale l'output sarà indirizzato.

Per rendersi conto della tecnica operativa dei predicati, vediamo insieme il seguente esempio:

```
GOAL:
writedevice(printer),write("Salve, MCMicrocomputer"),nl,writedevice(screen).
```

Se una stampante è accesa e collegata correttamente alla porta stampante [printer], il messaggio Salve, MCMicrocomputer sarà impresso su di essa; i successivi ordini reindirizzano le normali operazioni di I/O allo schermo. Se per un caso qualsiasi esiste un malfunzionamento (stampante non collegata o non accesa, stampante non pronta, ecc.) il sistema può bloccarsi o, magari, mostrare un messaggio d'errore con la solita richiesta «Abort, Ignore, Retry». Se le periferiche cui si decide di indirizzare le operazioni di I/O non sono file su disco, tutto quello che si è finora detto è sufficiente per manipolare le normali operazioni di I/O.

Al contrario le cose divengono più

complesse quando si tratta di lavorare su file.

Le operazioni di I/O su file

I programmi e le applicazioni che girano su una macchina possono conservare le informazioni che utilizzano direttamente nella memoria centrale o su memorie di massa; rappresentate oggi, nella maggior parte dei casi, da floppy o hard disk. In ogni caso, le informazioni vengono raccolte e conservate su file.

Come in altri linguaggi, esistono anche in Prolog due tipi fondamentali di file; i program-file e i data-file; nel primo caso la conservazione e il recupero sono affidati ai comandi save e load specifici del linguaggio (un programma è rappresentato da un unico gigantesco file sequenziale, della cui organizzazione interna ben poco interessa all'utilizzatore). La cosa davvero importante comunque è che quando viene salvato un file programma, i dati in esso contenuti sono del tutto persi (a meno che il programma stesso non contenga una routine per conservare i dati che possiede su un altro file, in questo caso un data-file). Per leggere da un programma dati contenuti in un data file, dati costruiti per servire e far funzionare correttamente il programma stesso, è necessario, all'inizio del programma, eseguire la direttiva «include» (delle direttive parleremo in una delle prossime puntate). La tecnica per leggere dati contenuti su un file presente sul disco attraverso un programma è descritta, passo passo, nella figura a. È importante ricordare ed eseguire l'ultimo passaggio, in quanto la mancanza di ridefinizione dell'I/O genera gravi errori, spesso non recuperabili, visto che il sistema non sa più dove inviare le relative operazioni di I/O dopo la chiusura del file stesso.

Dicevamo quindi della definizione del nome del file; la definizione avviene nella sezione [Domains], e il nome attraverso cui ci si riferisce è un nome simbolico, al contrario di quanto avviene comunemente in altri idiomi, in cui ci si riferisce al file con il suo nome fisico, con cui effettivamente è conservato sul disco. Niente impedisce, comunque, che il nome fisico e simbolico coincidano; un consiglio generale, ove non si adotti questa tecnica, è quello di usare come nome uno che, in qualche modo, somigli a quello fisico o ricordi quello che in questo è contenuto.

Il vantaggio di usare un nome simbolico è presto detto; tutto è legato alla limitazione di caratteri con cui i file vengono conservati sulla memoria di massa in MS-DOS. Così se Marinacci avesse, sul suo laptop i [lauti?!] compensi che fornisce al sottoscritto, potrebbe al massimo conservarli coll'acronimo «COMPDEM» o qualcosa di simile; molto più semplice assegnare un nome simbolico a tale file che potrebbe essere ridefinito come, che so, «compensi_faraonici_attribuiti_a_de_masi_il_sommo», tanto per esagerare e evidenziare l'utilità di tale notazione (per rendere onore al merito). Una seconda ragione, per non usare il nome fisico è che è possibile, attraverso nomi simbolici, ridefinire in diversi modi lo stesso file, cui poi accedere in maniera diversa a seconda delle circostanze; e non è finito, perché un nome simbolico aiuta a evitare confusione, impedendo che un accesso successivo allo stesso file avvenga secondo tecniche diverse, che potrebbero distruggere i risultati già in esso contenuti (si tenga conto che un'apertura indiscriminata in output su file può portare alla distruzione dei dati già contenuti sugli stessi). Perciò, visto che possiamo, cerchiamo di usare le comodità che la vita ci mette a disposizione.

La tecnica generale per la definizione del file è la seguente:

```
Domains
  file=saluti_alla_molinari
```

Il nome simbolico adottato sarà successivamente utilizzato per le operazioni di apertura, indirizzamento e chiusura del file. Ne parliamo nel prossimo paragrafo.

1 fornire un appropriato e compatibile nome simbolico al file destinato ad essere gestito dal programma, in modo che il programma stesso sappia come rintracciarlo sulla memoria di massa.
2 aprire il file secondo l'appropriata procedura di accesso
3 dichiarare, secondo quanto visto in (1) l'indirizzo di file su cui scrivere o leggere
4 usare il file
5 chiudere il file quando terminato
6 eventualmente ridefinire gli indirizzi di I/O attraverso (readdevice) e (writedevice)

Figura a - La procedura, passo dopo passo, per utilizzare un file dall'interno di un programma.

L'apertura dei file

Se apriamo il file semplicemente per leggere da esso, senza pretendere di modificare alcunché, il predicato [openread] è quello che fa per noi; esso maneggia due argomenti: il nome simbolico assegnato al file e, ovviamente, il nome fisico presente sul disco, cui il nome simbolico si riferisce. La sintassi generale dell'istruzione è:

```
Clauses
  openread(elenco_telefonico,"ELTEL")
```

Questa operazione sostituisce la tastiera con il file DOS ELTEL e lo rende il corrente indirizzo di input, fino a quando un nuovo predicato non renderà di nuovo la tastiera la corrente periferica di

• dichiarazione del file nella sezione (domains)
• uso del predicato (openmodify) per aprire il file
• uso di (readdevice) per indirizzare le corrette procedure di lettura
• uso di una funzione di lettura per richiamare dati in memoria
• uso di (readdevice) per riassegnare il controllo alla tastiera
• modifica dei dati come necessario
• uso di (writedevice) per reindirizzare i dati sul file
• uso di (writedevice) per riportare l'output sullo schermo
• eventualmente, ritornare daccapo.

Figura b - Il processo di modifica e verifica di un file.

input. Prolog rispetta la normale nomenclatura DOS, che potrà includere eventualmente il nome del drive o il pathname della directory, per la corretta individuazione della «locazione» del file, che così potrà non risiedere necessariamente sullo stesso disco su cui insiste il programma.

Le virgolette attorno al nome fisico del file sono necessarie solo in due casi; se si usano, per la prima lettera, caratteri maiuscoli (Prolog immaginerebbe che si tratta di una variabile, e cercherebbe di interpretare la stessa invece di considerarla come vero nome), e se esistono, nel nome punti (quando ad esempio si desidera specificare il suffisso del nome stesso). In ogni caso è possibile usare caratteri minuscoli anche se il nome del file DOS è in maiuscolo.

Se, al contrario, desiderassimo aprire il nostro file per modificare il contenuto del file stesso, occorre aprire il file con un predicato che informi Prolog sul tipo di modifica che desideriamo venga eseguita; i tipi di modificazione sono i seguenti:

- scrittura - creare un file o sovrascrivere uno già esistente, cancellando l'intero contenuto sostituendolo con nuove informazioni;
- appending - aggiungere dati alla fine del file già esistente;
- modifica - cambiamento del contenuto di parte del file senza togliersi la possibilità di aggiungere nuove informazioni.

Con una mnemonica abbastanza facile da intendere, Prolog utilizza, per queste tre funzioni tre predicati appropriatamente chiamati [openwrite], [openappend] e [openmodify]. Tutti e tre i predicati maneggiano, in analogia a [openread] due argomenti; il nome simbolico assegnato al file e il nome DOS. Un esempio delle tre funzioni applicate all'operatore utilizzato con [openread] sarebbe:

```
Clauses
  openwrite(elenco_telefonico,"ELTEL")
Clauses
  openappend(elenco_telefonico,"ELTEL")
Clauses
  openmodify(elenco_telefonico,"ELTEL")
```

L'ultimo predicato come è facile intuire, è il più elastico e efficiente dei tre, e sovente può sostituire gli altri due con una maggiore snellezza delle procedure del programma; esso infatti permette di intervenire selettivamente su parti ben determinate del file, eseguire le necessarie modifiche, senza mai escludere la possibilità di aggiungere alla fine nuove informazioni. Alla fine sarà sufficiente chiudere il file e riaprirlo in lettura [openread] per verificare la qualità delle modifiche effettuate.

La figura b, mostra in dettaglio le fasi operative descritte nel periodo precedente; come si vede si tratta di operazioni molto intuitive, che non abbisognano di grandi spiegazioni. L'uso abbondante fatto delle funzioni [readdevice] e [writedevice] è necessario per poter utilizzare tastiera e schermo alla bisogna.

Una volta padroneggiata la funzione [openmodify] le altre due sono più facili da usare in quanto ognuna di esse è solo parte delle altre. Ciononostante occorre prestare attenzione nell'uso dei predicati [open...] in quanto possono accadere cose diverse, in presenza o meno di un file chiamato dal disco, secondo quanto si vede nella figura c.

Data la pericolosità di certe operazioni su file, Prolog fornisce uno speciale predicato [existfile] che permette di verificare se un file DOS (non un nome simbolico, ovviamente) esiste su disco. Questo predicato setta una variabile booleiana a TRUE o FALSE, a seconda della circostanza e del successo della ricerca. Non mi pare di aver visto un comando analogo in altri linguaggi; certo è che può salvare da perdite disastrose di dati se si crea un nuovo file senza rendersi conto che ne esiste già uno con questo nome.

E per concludere con questo argomento vediamo come chiudere i file. Sebbene il linguaggio provveda al completamento delle operazioni previste dal programma, a chiudere automaticamente tutti i file aperti, il predicato [closefile] permette di lavorare con maggior ordine e con quel pizzico di savoir-faire che non guasta anche in una room di programma; d'altro canto tentare di chiudere un file già chiuso o magari non esistente non crea alcuna condizione d'errore; perché non stare tranquilli?

Anche [closefile] accetta i soliti due argomenti, come i predicati visti precedentemente; la loro collocazione e il loro significato è del tutto intuibile.

E anche stavolta abbiamo completato con la trattazione dei file; ci resta da trattare l'accesso ai file [random] e ad altre periferiche (schermo e finestre); ne riparleremo la prossima volta.

PREDICATO	FILE TROVATO	FILE NON TROVATO
[openread]	apre un file per lettura	ERROR
[openmodify]	apre un file per modifica	crea il file
[openappend]	apre un file per modifica	ERROR
[openwrite]	CANCELLA E SOVRASCRIVE SUL FILE	crea il file

Figura c - Interazione tra i predicati «open»e i file presenti sul disco.

A500 + ESPAN L. 758.000

ies COMPUTER

SERVIZIO ASSISTENZA

06/3651588 **06/3651688**

XT 8088,10 MHZ,1 FD 360,1 HD 20 MB, 512KB, TASTIERA,SK MONOCR	L. **945.000**
AT 286,16 MHZ,1 FD 1.2,1 HD 20MB, 1024KB, TASTIERA,SK MONOCR	L. **1.390.000**
386 25 MHZ,1 FD 1.2,1 HD 40MB, 1024KB, TASTIERA, SK VGA	L. **2.800.000**
HD 20MB	L. **310.000**
HD 40MB (veloce 19 m/s)	L. **750.000**
HD 80MB (veloce 19 m/s)	L. **1.370.000**
DRIVE 1.2MB	L. **160.000**
DRIVE 720KB	L. **130.000**
DRIVE 1.44	L. **160.000**
CONTR. HD XT	L. **89.000**
CONTR. AT	L. **150.000**
SK HERCULES	L. **70.000**
SK CGA	L. **70.000**
SK DUAL	L. **85.000**
SK EGA	L. **145.000**
SK VGA (256 kb)	L. **275.000**
SK SUPER VGA (512 kb,16 bit)	L. **320.000**
MONITOR	
MONOCROMATICO BIFREQUENZA	L. **163.000**
VGA MICROVITEC	L. **750.000**
MULTISINC	L. **850.000**
FAX CANON	
FAX 80	L. **1.250.000**
FAX 120	L. **1.590.000**

PREZZI IVA INCLUSA

18 MESI DI GARANZIA SU TUTTI I PC

STAMPANTI EPSON

LX 800	L. **400.000**
LQ 500	L. **600.000**
FX 1050	L. **980.000**
LQ 1050	L. **1.360.000**
GQ 5000	L. **3.000.000**

STAMPANTI CITIZEN

120 D	L. **310.000**
180 E	L. **330.000**
15 E	L. **530.000**
SWIFT 24	**Telefonare**
PRODOT 9	**Telefonare**
PRODOT 9X	**Telefonare**

EPSON

CITIZEN