di Raffaello De Masi

Le operazioni di I/O

L'input-output è l'operazione più potente e la vera necessaria in un calcolatore. Ragionando per assurdo, un calcolatore che non sia capace di ricevere dall'esterno segnali, né di inviare a esso risposte è praticamente inutile (per uno di quegli scherzi del caso, mi trovo a parlare, nella rubrica di Intelligenza Artificiale, proprio di interfacciamento dei SE con l'esterno)

Dobbiamo perciò mettere in condizione la macchina di parlare, colloquiare, scambiare dati e perché no, visto che stiamo trattando di linguaggi intelligenti, pareri; si tratta, per certi versi, di una operazione abbastanza semplice da implementare e da pilotare, ma, al contrario, di difficile organizzazione in questa particolare branca dell'informatica, dove, a un linguaggio, non viene solo richiesto di eseguire manipolazioni e dare risposte, ma anche e soprattutto di colloquiare in maniera continua, e talora in tempo reale, con l'esterno. Tutto ciò comporta un'ardua manipolazione delle routine software e probabilmente, il Prolog è il linguaggio che più di tutti, nel campo dell'I/O, è chiamato a operazioni di notevole complessità e di avanzate prestazioni.

Anche qui, ovviamente, ci intratterremo in particolare sulle tecniche di base del linguaggio per due motivi; primo perché, successivamente, quando parleremo in maniera più avanzata e specifica del Turbo Prolog della Borland, vero campione in campo, avremo modo di analizzare appieno gli operatori disponibili; secondo perché la trattazione generale da cui ben poche volte ci siamo discostati finora ci impone di generalizzare assolutamente il discorso relativo agli operatori; questo, unitamente al fatto che abbiamo più volte menzionato, che forse mai come nel Prolog i discorsi di uniformazione e standardizzazione del linguaggio cadono nell'assoluto vuoto e indifferenza, porta a ridurre a ben pochi gli operatori qui descritti, che per forza di cose divengono assolutamente standard e molto simili a quelli presenti in altri linguaggi.

Questa precisazione era d'altro canto doverosa, visto che, in una discussione avuta con un collega analista, col quale sto costruendo un sistema esperto destinato al riconoscimento delle micosi, mi veniva fatto notare che quanto avevo finora esposto in queste mie discussioni sulle pagine della rivista, non compariva molto di più di quello che era, appunto, disponibile in altri linguaggi. Vero! Ma non si possono servire contemporaneamente più padroni; tanto per intenderci, se si adotta il Turbo Prolog (giunto in versione 2) gli operatori di stringa di cui abbiamo parlato la vostra scorsa si triplicano addirittura; ma d'altro canto sono convinto che a questa conclusione erano già giunti i miei lettori (che spero rumoreggianti in masse oceaniche anche se mi sfiora il dubbio che non siano più numerosi di una qualsiasi armata Brancaleone!), visto che gli operatori descritti, ad esempio, la volta passata, sono debolucci come quelli del più scalcinato Basic o Pascal disponibile su uno straccio di Commodore 64 o di un Amiga 500.

Comunque ritorniamo al nostro discorso! Dicevamo di I/O; Prolog, sebbene disponga nelle sue diverse implementazioni di raffinatissime tecniche a tal proposito (si tratta dell'area di programmazione forse più curata e avanzata del linguaggio, e ne è ovvio il motivo), non è comunque interattivo in tempo reale, appannaggio, questo, di linguaggi ben più dotati e efficienti in tal senso; d'altro canto, a costo di ripeterci, diremo che non si tratta di una esigenza propria del nostro idioma, i cui compiti non sono certo quelli di gestire strumentazione, macchine o robot; lasciamo a questi compiti linguaggi come il Forth o l'Assembler, molto più dotati.

Gli operatori di I/O

Come d'altro canto avviene in molti altri linguaggi, anche in Turbo Prolog sono disponibili una serie di predicati di diversa natura, destinati a manipolare dati provenienti dalla tastiera o da file su memoria di massa, e capaci di dare risposte o ridepositare sulla memoria di massa stessa i dati rielaborati alla bisogna.

Cominceremo, nella nostra trattazione, a parlare di operatori standard per poi passare a quelli specializzati e, probabilmente, più specifici per il linguaggio.

Gli operatori standard di input

Tutte le operazioni di output in Prolog sono maneggiate e organizzate dal predicato [write]. Attraverso di esso, ovviamente utilizzando opportuni indicatori e indirizzatori, è possibile dirigere le operazioni di output a CRT, stampanti e file, o a qualunque altro indirizzo (ad esempio una macchina di registrazione o, come accade sovente nei SE

medici, veri campi d'azione dell'Intelligenza Artificiale, ad attrezzature di analisi e controllo). Al contrario della maggior parte dei predicati Prolog, [write] può maneggiare un numero imprecisato di argomenti; il principio generale è che esso semplicemente mostra o stampa tutti gli argomenti che lo seguono.

L'esempio più semplice dell'uso di [write] è quello di fargli scrivere una stringa sullo schermo; questo è l'indirizzo di default a meno che non ci siano specificatori (li vedremo tra poco) che indirizzano le operazioni a un'altra periferica.

Un esempio dell'operazione è:

```
Goal: write("hello!")
hello! True
Goal:
```

La forma generale, come si vede, è rappresentata dall'operatore [write] seguito dalla stringa che si intende stampare: la lunghezza della stringa può essere qualsiasi, e, proprio in base a quanto detto qualche secondo fa, può essere rappresentato da più stringhe; la cosa è importante specie se si tiene conto che ci sono casi in cui occorre mischiare valori numerici e alfanumerici tra di loro, evitando la seccatura della conversione stringa-numeri e viceversa che abbiamo visto la volta passata. Un esempio può essere rappresentato da:

```
Goal: write("Salve","Marinacci!")
Salve,Marinacci!True
Goal:
```

o, ancora

```
Goal: write("Ho vinto" ",4," miliardi alla lotteria di Capodanno")
Ho vinto 4 miliardi alla lotteria di CapodannoTrue
Goal:
```

Un altro operatore abbastanza comune, abbinato a [write] è [nl], dal significato esattamente eguale a quello che ha in «C», [new line]; esso ci consente di dare un aspetto più pulito e professionale al nostro output che sarà del tipo:

```
Goal: write("Finora la Molinari non ha ancora telefonato, meno male!") and nl
Finora la Molinari non ha ancora telefonato, meno male!
True
Goal:
```

E se volessimo spezzare la stringa in più parti? Semplice (ma non troppo); sarebbe sufficiente battere:

```
Goal: write("Finora la Molinari non ha ancora telefonato,")
Goal: write ("meno male!") and nl
Finora la Molinari non ha ancora telefonato,
meno male!
True
Goal:
```

Ma c'è qualcosa che non funziona; primo che faticaccia per scrivere qualcosa, secondo, in maniera interattiva, vale a dire direttamente dalla tastiera, la cosa non funzionerebbe di certo, visto che dopo il primo goal la risposta sarebbe immediata; avremmo, in pratica due ordini e due risposte, spezzettati nel tempo e ben poco pratici per la verità; è possibile rendere tutto più compatto ed elegante in maniera chiara e pratica adottando una stringa di formattazione inserita nella stringa stessa, [\ n], anche stavolta del tutto simile come significato all'equivalente in «C». Così avremo:

```
Goal: write ("Buongiorno, \ n Giovanna! \ n).
Buongiorno,
Giovanna.
True
Goal:
```

C'è da notare che il linguaggio considera l'operatore [\ n] come una stringa diversa da qualunque cosa le sia accanto; così mentre è possibile battere

```
Goal: write(salve!)
salve!True
Goal:
```

senza le virgolette, è invece illecito digitare:

```
Goal: write(salve \ n)
```

cosa che darebbe un segnale d'errore (in Turbo Prolog [error 10]).

[write], come prevedibile, non è solo utilizzabile per scrivere e mostrare stringhe preformattate; di esso ci possiamo servire per scrivere variabili, che possono essere, ovviamente, mischiate a stringhe, e così via.

Ancora un esempio in figura 1 e in figura 2 un altro più complesso.

Utilizzando il semplice predicato [write], l'output non si presenta formattato in maniera particolarmente raffinata; avremo in uscita una stringa standard,

```
Goal: Variabile 1 = "esempio"
variabile 1 = esempio
True
Goal: write("Questo è un Variabile1", "di output sullo schermo \ n")
Questo è un esempio di output sullo schermo.
Goal:
```

Figura 1

```
Goal: Variabile 1 = "esempio", Variabile 2 = "più complesso", write ("Questo è un", Variabile 1, Variabile 2, "di output sullo schermo. \ n").
Questo è un esempio più complesso di output sullo schermo.
Variabile 1 = esempio
Variabile 2 = più complesso
1 Solution
Goal:
```

Figura 2

giustificata a sinistra, e di grandezza normale; ma talvolta può essere opportuno scrivere informazioni su periferiche grafiche o su disco, opportunamente formattate; in questo caso può essere opportuno far ricorso a un predicato più raffinato, [writef].

Come il suo più diretto predecessore, [writef] (letteralmente write-formatted) maneggia un numero indefinito di argomenti, ma il primo ha un significato e un valore particolare e speciale; la prima lettera deve essere il segno del [%], la seconda è rappresentata dal segno meno [−], solo se si desidera che la stringa sia giustificata a destra; la mancanza di tale segno (o il segno [+] in talune implementazioni), determina il normale incolonnamento a sinistra.

Il campo successivo, opzionale, specifica la minima grandezza di campo da utilizzare nella stampa, su schermo o su stampante, di quanto contenuto nella lista di argomenti maneggiati dalla funzione (predicato) stessa.

Ancora (ma non ci si spaventi per questo, è più complicato da dire che da fare), segue un punto [.] e un numero (o un gruppo da una a tre lettere).

Nel primo caso (numero) la cifra indica il numero massimo di lettere da stampare prelevato dalla stringa stessa (o, nel caso di output numerico di numero reale, il numero (precisione) delle cifre dopo il punto decimale).

Una lettera (che può essere «f», «e», o «g») presente in ultima posizione significa una serie di cose. Il parametro [f] (floating) avvisa il sistema di mostrare i numeri nella forma fissa decimale, vale a dire nel formato standard per il linguaggio; [e] (exponential) mostra i valori in forma esponenziale, e infine [g] lascia la scelta, a Prolog, di mostrare i risultati nella maniera migliore e più consona ai quantitativi numerici manipolati al momento.

Gli operatori standard di output

L'istruzione [write], con le sue diverse implementazioni e variazioni sul tema, può manipolare in maniera molto elastica diversi tipi di dati (caratteri, stringhe, stringhe concatenate, simboli, variabili, interi, e numeri reali) anche variamente frammisti tra loro. La sua controparte in input, il predicato [read] non gode di tutta questa libertà, e deve essere esplicitamente confezionato in relazione all'informazione che è destinata a maneggiare; in altre parole deve conoscere il tipo di dati che dovrà maneggiare.

Il predicato [read], in Turbo Prolog, ha cinque variazioni. La sua variazioni più flessibile è [readln] (ma che somiglianza col «C»!), che legge semplicemente caratteri dalla periferica di input corrente fino a incontrare una virgola o un CR (la periferica standard di input, ovviamente, è, in default, la tastiera).

Un esempio di uso di [readln] in fase interattiva è:

```
Goal: readln(Frase).
```

A questo punto, dopo aver premuto il [return] non succede proprio niente; ma è un niente diverso dalla solita fase di stasi della macchina.

In questa, infatti, è sempre presente, all'inizio del rigo, il [Goal] qui il sistema mette in fase di attesa, e il cursore si pone all'inizio della linea successiva, lampeggiando in attesa. Battiamo:

```
Questo è un articolo per
MCmicrocomputer
```

Il sistema si risveglia e prosegue con:

```
Questo è un articolo per
MCmicrocomputer
Frase = Questo è un articolo per
MCmicrocomputer
1 Solution
Goal:
```

Facciamo una piccola variazione sul tema, battendo:

```
Goal: readln (Frase).
Questo è un articolo, per
MCmicrocomputer
Frase = Questo è un articolo
1 Solution
Goal:
```

E l'articolo [per MCmicrocomputer]? Chissà dove è andato a finire, perso nei meandri infiniti e tortuosi della macchina! Molto più semplicemente è sparito, in quanto la variabile di input è stata soddisfatta direttamente dalla prima parte della frase (si noti anche la scomparsa della [.]). Battiamo ancora:

```
Goal:readln (Frase 1, Frase 2).
Questo è un articolo, per MCmicrocom-
puter
Frase 1 = Questo è un articolo
Frase 2 = per MCmicrocomputer
2 Solutions
Goal:
```

Questo è solo una delle tipologie di [read]; esistono altre quattro forme di input. [readchar] legge dalla tastiera un solo carattere e, come i soliti Inkey$ et similia di altri linguaggi è utile essenzialmente in operazioni di risposta [si-no] o per fermare l'attività di un programma. Un esempio potrebbe essere:

```
Goal: write ("batti un tasto:"),
readchar(Tasto), nl,write ("Hai battuto il
tasto ",Tasto,nl).
batti un tasto: a (battuto dalla tastiera)
Hai battuto il tasto a
Tasto = a
1 Solution
Goal:
```

Per leggere numeri Prolog mette a disposizione due predicati, praticamente simili, ma destinati a maneggiare forme numeriche diverse, [readint] e [readreal], dal significato abbastanza ovvio; il primo legge numeri interi, il secondo numeri reali. L'unica attenzione da porre nell'uso di questi predicati è che, ovviamente [readint] non può leggere numeri decimali (ma il contrario è possibile).

Il tutto è abbastanza chiaro negli esempi successivi.

```
Goal; readint(Numero).
55
Numero = 55
1 Solution
Goal; readint (Numero).
55.12
No Solution
Goal; readreal (Numero).
55.36
Numero = 55.36
1 Solution
Goal; readreal (Numero).
55
Numero = 55
1 Solution
```

Questo perché, è appena il caso di accennarlo, tutti i numeri interi sono anche numeri reali.

E per finire accenniamo all'ultimo predicato di input [readterm]. Il suo uso non è interattivo, essendo riservato a leggere dati conservati su file presenti su disco.

Tramite esso è possibile accedere ad una unità di registrazione presente in un file; esso maneggia due argomenti; il primo stabilisce il dominio dell'oggetto da leggere, il secondo il nome della variabile in cui il valore verrà riposto; la sua forma tipica è:

```
readterm (a,B)
```

Bene, siamo di nuovo alla fine; ma anticipiamo, come al solito, l'argomento del prossimo numero.

Nella discussione abbiamo più volte precisato che la periferica standard, di default, è la tastiera; ma come si fa ad accedere alle altre? E come si fa a lavorare su file, che a tutti gli effetti possono essere considerati come vere periferiche?

MC

ELETTRONICA CENTOSTELLE s.r.l.

ZENITH	Lap top
TANDON	Desk top
ASEM	Desk top
NEC	Stampanti

Via Centostelle, 5/a - Firenze - Telefono (055) 61.02.51 - 60.81.07 - Fax 61.13.02

SOFTWARE

WORD PROCESSOR

Microsoft Word 5	it. L.	712.000
Microsoft Word 5 euro	it. L.	570.000
MicroPro Wordstar Prof. 4.0	it L.	590.000
MicroPro Wordstar prof. 5.0	in L.	590.000
MicroPro Wordstar 2000 3.0	it L.	880.000
Lotus Manuscript 2.0	in L.	730.000
AshtonTate Multimate adv. II	it L.	785.000
Borland Sprint	in L.	330.000
Word Perfect	it L.	960.000

SPREADSHEET INTEGRATI

Microsoft Excel	it L.	712.000
Microsoft Excel Euro	in L.	640.000
Microsoft Works	it L.	280.000
Lotus 1 2 3	it L.	695.000
Lotus Symphony	it L.	890.000
Ashton Tate Framework III	it. L.	952.000
Borland Quattro	it L.	330.000
Computer Ass. Supercalc 5	it L.	800.000

DATA BASE MANAGEMENT

AshtonTate dBase III plus	it L.	880.000
AshtonTate dBase IV	it L.	1.080.000
AshtonTate Rapid file	it L.	560.000
Borland Paradox	it L.	1.080.000
Borland Paradox (os/2)	it L.	1.400.000
Borland Paradox 386	it L.	1.400.000
Borland Reflex 2.0	it L.	340.000

GRAPHICS

Microsoft Chart 2	it L.	390.000
Microsoft Chart 3 euro	in L.	540.000
Lotus Freelance V.2.01	it L.	650.000
Auto cad 10.0 (scuole università)	it L.	1.190.000
Paintbrush plus (per Wind.)	in L.	240.000
Gem Artline	in L.	1.350.000
Gem desktop publishers	in L.	650.000
Lotus GraphWriter II	it L.	723.000
Adobe Illustrator	in L.	1.390.000

DESKTOP PUBLISHING

Ventura Publisher	it L.	1.480.000
Fonts Bitstream	it L.	550.000
AshtonTate Byline	it L.	472.000

AMBIENTI OPERATIVI

Microsoft Project 3.0	it L.	760.000
Microsoft Project 4 Euro	in L.	680.000
Microsoft Windows 286	it L.	180.000
Microsoft Windows 386	it L.	280.000
Microsoft Windows 286 toolkit	in L.	680.000
Lotus Agenda	in L.	650.000

LINGUAGGI

Microsoft Quick basic 4.5	in L.	145.000
Microsoft Quick C compiler	in L.	145.000
Microsoft Basic Compiler 6.0	in L.	380.000
Microsoft C Compiler 5.1	in L.	590.000
Microsoft Fortran Compiler	in L.	630.000
Microsoft Cobol Compiler V3	in L.	1.100.000
Microsoft Macro Assembler	in L.	240.000
Microsoft Pascal Compiler	in L.	550.000
Microsoft OS/2 toolkit	in L.	480.000
Borland turbo Pascal 5.5	it. L.	240.000
Borland turbo basic	it L.	170.000
Borland turbo C 2.0	it L.	240.000
Borland turbo Prolog. 2.0	it L.	230.000
Borland turbo Assembler/debug	it L.	290.000
Borland turbo C professional	it L.	399.000
Borland turbo Pascal Profess.	it L.	399.000
RM Cobol 85	in L.	2.765.000
RM Cobol Compiler	in L.	1.660.000
RM Fortran	in L.	1.405.000

UTILITIES

Norton Utilities	in L.	170.000
Norton Commander	in L.	170.000
PC Tools 5.5	in L.	150.000

HARDWARE

Hard Disk Nec 5" 1/4 20 Mb + control	L.	566.000
Hard Disk Seagate 5" 1/4 40Mb + control.	L.	857.000
Copr. Metem. 8087/5 4.77 MHz	L.	233.000
Copr. Matem. 80287/8 8MHz	L.	461.000
Copr. Matem. 80287/10 10 MHz	L.	475.000
Copr. Matem. 80287/12 12 MHz	L.	585.000
Copr. Matem. 80387/16 16 MHz	L.	840.000
Copr. Matem. 80387/33 33 MHz	L.	1.455.000
Modem int. Hyundai V 21:22	L.	119.000
Modem est	L.	189.000
Monitor colori 14" Ega	L.	748.000
Monitor colori Multisink NEC	L.	1.193.000
Monitor Multisink fosfori bianchi	L.	312.000
Scheda grafica ris. VGA	L.	490.000
Espansioni RAM		Telefonare
Compact Disk Hitachi		Telefonare
Stampante portatile Toshiba	L.	699.000
Stampante 24 aghi 80 col	L.	796.000
Fax Toshiba T 211		Telefonare
Scanner IOR		Telefonare

NOVITÀ

Microsoft Quick basic 4.5	it L.	195.000
Microsoft Quick pascal 1.0	in L.	155.000
Microsoft Quick MASM/	in L.	225.000

Concessionario **TOSHIBA**

Per lo studente
Toshiba T 1000
Stampante NEC 24 aghi P 2200
Software WORKS (Microsoft)

Per il professionista
Toshiba T 1200 con HD 20Mb
Stampante STAR 24 aghi 80 col.
Programma di videoscrittura
WORD (Microsoft) in italiano

Per le università, scuole e istituti
Toshiba T 1600
Coprocessore matematico
Autocad 10.0

TOSHIBA T1000 SE
ancora più piccolo
ancora più potente

Tutti i prezzi sono IVA esclusa
Pagamento in contrassegno, vaglia o VISA
Per ordini inferiori a L. 500.000
aggiungere spese postali L. 10.000

ORDINI
a mezzo telefono
Fax
Posta

Consulenza telefonica
gratuita su tutta la
nostra gamma di prodotti
Inserimentio automatico dei nostri clienti
nel servizio Direct Marketing

SI RACCOMANDA AL MOMENTO DELL'ORDINE DI CHIEDERE CONFERMA DEI PREZZI