di Raffaello De Masi

Elementi di Prolog

quarta parte

Lo sviluppo di un programma in Prolog

Dopo aver dato uno sguardo d'insieme alla struttura generale di un programma, è opportuno analizzare più da presso le caratteristiche di disegno e sviluppo di un programma in Turbo Prolog, attraverso l'esplorazione delle tecniche di redazione e di utilizzo di un programma. La cosa è abbastanza semplice se si considera che la vera fase di programmazione, in Turbo Prolog, è quella di definire, nel modo più corretto possibile, gli oggetti, le regole, e le relazioni che intercorreranno tra esse. Un corretto e ben disegnato «shell» (se così si può dire) comprendente questi particolari è la migliore garanzia di rapida efficienza, veloce debug, e buona leggibilità del programma stesso. D'altro canto nessun essere pensante, sia uomo o macchina, sarebbe capace di estrarre concetti e conoscenze da una struttura non razionale o dotata di conoscenze smozzicate. Perciò, ordine e disciplina, in Prolog come nella vita, se vogliamo chiarezza di intenti e velocità di risultati

Dan Shafer, nel suo eccellente volume «Turbo Prolog Primer», che rappresenta parte della bibliografia essenziale per chi desidera affrontare in maniera completa e professionale lo studio di questo linguaggio, evidenzia, a pagina 85, che, nel caso del linguaggio in esame, non è il caso di parlare di base di dati, ma di base di conoscenze, termine, d'altro canto, comunemente usato in Intelligenza Artificiale. La differenza non è peregrina ed accademica, se si considera che una base di dati è una raccolta arida e del tutto disarticolata di dati messi insieme, per necessità di cose, in base ad un unico denominatore o finalità comuni. Una base di conoscenze, invece, non solo elenca oggetti, dati, azioni, ma descrive anche le relazioni intercorrenti tra essi, vale a dire che presenta anche le *«regole»* che intercorrono tra gli elementi. Appare pertanto evidente che il programmatore in Prolog parte col piede buono se ha come principale intendimento quello della descrizione esatta, non opinabile né fraintendibile, degli oggetti che descriverà e delle interconnessioni relative.

I fatti

Come già ricorderete dalla volta scorsa l'elemento di base in un programma in Prolog è il blocco dei «fatti», elementi che possono essere riassunti in tre regole principali:

— uso e significato delle lettere maiuscole-minuscole
— posizione degli elementi nella descrizione del fatto
— uso e destinazione della punteggiatura.

La prima regola impone di stare attenti al tipo di carattere; generalmente, in Prolog, l'uso delle lettere maiuscole è riservato a particolari scopi; ragion per cui, almeno per adesso, va generalizzato l'uso nelle lettere minuscole; così un fatto, nella sua forma più generica può essere così codificato:

```
relazione (oggetto,oggetto).
```

e, esemplificando

```
preferisce (corrado,ibm)
```

Come si vede, il nome del «sapiente» di MC viene scritto tutto in minuscole; scrivere [Corrado] stravolgerebbe il senso del «fatto», come vedremo successivamente quando parleremo delle variabili e degli oggetti non definiti.

Per caso o per volontà di chi scrive l'esempio contiene le tre sole forme di punteggiatura più comuni utilizzate in Turbo Prolog, rappresentate dalla parentesi tonda [(], aperta e chiusa, dalla virgola [,] e dal punto fermo [.]. Ne vedremo, tra poco, i significati, anche se sono abbastanza intuibili. Ci preme adesso evidenziare la struttura della frase.

La relazione che lega tra loro gli oggetti [preferisce] è sempre il primo elemento nella descrizione di un fatto, qualunque sia il numero ed il tipo degli oggetti coinvolti dalla relazione stessa. Pertanto espressioni come

```
comanda (marco,technimedia)
comanda (mia_moglie,casa)
accompagna (antonio,andrea,scuola)
```

sono tutte valide. Gli oggetti manipolati dalla relazione iniziale sono compresi tra parentesi e separati da virgole. Detta in altri termini, la regola può essere così semplificata:

```
predicato_verbale (soggetto, oggetto, [com-
                   plemento]...)
```

e rappresenta una situazione ideale anche per venire incontro alle necessità

del compilatore, che sa che, in ogni fatto, la prima parola rappresenta l'azione che viene eseguita, la seconda il soggetto che esegue l'operazione, e così via. Questo modo un po' inusuale di presentare le cose diviene, dopo poco tempo, familiare all'utente, che non incontra più soverchie difficoltà nella comprensione di cosa fa effettivamente il «fatto».

In gergo, la notazione viene indicata come tecnica di rappresentazione prefissa e, prima che in Prolog, ha illustri precedenti in LISP. È esattamente il contrario di quanto avviene, ad esempio, in Forth, dove domina la notazione postfissa, o nella notazione polacca inversa, almeno nel campo aritmetico.

Infine occorre parlare della punteggiatura che, al contrario di quanto avviene in Pascal, è del tutto intuitiva e, comunque, estremamente semplificata. Le notazioni principali sono, come abbiamo già detto, tre: il punto, la parentesi e la virgola. Quest'ultima serve essenzialmente a separare elementi dello stesso tipo o compresi nello stesso modello; le parentesi hanno un uso intuitivo, come si vede, e non necessitano di soverchie spiegazioni tranne per il fatto che è possibile che livelli diversi di parentesi siano nidificati, quando un fatto è compreso come argomento di un altro (e così via).

Il punto termina logicamente e fisicamente un comando. Esso è d'obbligo alla fine di un fatto, tranne nel caso in cui, omettendolo, il fatto stesso è seguito da un supporto condizionale [if]. Inoltre è consentito, in Turbo Prolog, omettere il punto al termine di un "goal", e, in questo caso, questo linguaggio è fuori standard.

In questo modo abbiamo costruito una piccola base di conoscenza (ancorché rappresentata da una sola relazione). Occorre però, in base alla gerarchia definita nelle puntate precedenti, dichiarare i tipi di nome e simbolo nella sezione "Domain". Come abbiamo già visto la volta scorsa, è possibile definire interi, stringhe, simboli, liste. Delle differenze intercorrenti tra questi elementi abbiamo già discusso a lungo le volte scorse; ci preme soprattutto ricordare che simboli e stringhe sono, per tutti gli usi pratici, normalmente intercambiabili. Ricorderemo che le stringhe hanno bisogno di essere definite sempre dalla doppia virgoletta ["] all'inizio ed alla fine, mentre nel secondo caso esse sono necessarie solo quando la presenza di spazi bianchi genererebbe confusione nel compilatore. Una differenza sottile

```
Domains
        persona,auto = symbol.

Predicates
        guida(persona,auto).
        preferisce(persona,auto).
        regala(persona,auto).

Clauses
        guida(corrado,bmw).
        guida(massimo,lotus).
        regala(marco,alfa_romeo).
        preferisce(giovanna,volvo).
        regala(giovanna,lancia).
        guida(corrado,digital).
        preferisce(massimo,pompelmo).
```

Figura A

ma invisibile all'utente ordinario è che i simboli sono maneggiati meglio dal compilatore ma occupano maggior spazio in memoria, mentre con le stringhe il compilatore è afflitto da un maggior lavorio. Questione di gusti o di opportunità ma se si considera il basso costo della memoria oggi credo che la scelta sia abbastanza semplice.

Le clausole vanno sempre definite, inizialmente attraverso un predicato; cosa è un predicato? Semplice, è il canovaccio di una clausola; se vogliamo, in termini più semplici, un modello dell'istruzione (di conoscenza) che intendiamo fornire al compilatore. Avanti ancora una volta col solito esempio:

preferisce (marco,olivetti)

è una clausola; facciamola procedere da

preferisce (persona,calcolatore)

abbiamo creato una maschera d'uso della clausola; vale a dire che abbiamo indicato al calcolatore che l'istruzione «preferisce» manipolerà due symbol; allora differenzieremo le due righe nel modo seguente:

```
Predicates
        preferisce (persona,calcolatore)
Clauses
        preferisce (marco,olivetti).
```

(notare punteggiatura e simbologia). Ma per completare dobbiamo definire la natura del «materiale» maneggiato da [preferisce]. Allora scriviamo qualcosa di più complesso in un nuovo esempio mostrato nella figura a), di cui si noti sempre la punteggiatura.

Si tratta di un discorso articolato e preciso, dove accanto ad una dichiarazione iniziale del tipo di «materiale» che

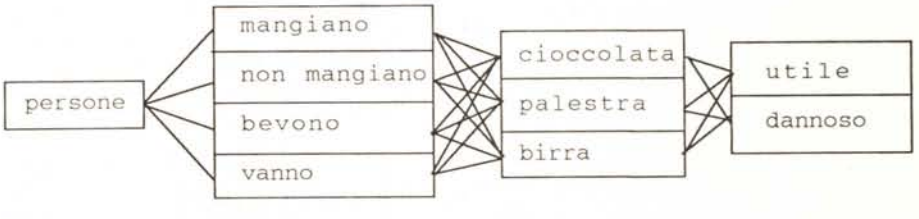


Figura B

	cioccolata	birra	palestra
andrea	mangiare	- -	- -
biagio	mangiare	- -	- -
carlo	mangiare	- -	- -
diana	mangiare	- -	- -
ernesto	- -	- -	- -
francesco	- -	- -	- -
giovanna	- -	- -	- -
italo	- -	- -	andare
luana	- -	- -	andare
marcello	- -	- -	andare
nora	- -	bere	- -
olga	- -	bere	- -
pasquale	- -	bere	- -

Figura C

la relazione manipolerà, viene fornito [Predicates] un modello operativo della regola, e successivamente viene trasmessa una base di conoscenza, anzi per essere precisi, ben tre basi articolate sulle relazioni [guida], [preferisce] e [regala]. Notare che ogni clausola ha lo stesso formato del predicato immediatamente prima definito (a meno che, ovviamente, il formato non sia stato «costruito» in una dichiarazione di predicato posta in una precedente parte del programma).

Bene, abbiamo, anche se nel nostro piccolo, costruito una base di conoscenze organizzata, un mondo, comprensibile dalla macchina, in cui giovanna viaggia in volvo e regala, bontà sua, automobili lancia ai suoi ammiratori, corrado guida le digital (ovviamente la macchina non è capace di distinguere assurdi tecnologici; in fondo il Prolog è un bonaccione, crede a tutti) e massimo sta facendo, probabilmente una cura dimagrante. Si vede quindi che maggiore è la quantità di informazioni che si aggiungono al costrutto delle clausole, migliore sarà la conoscenza del mondo delle bmw e delle lotus da parte della macchina; infatti, dalla figura a la macchina non può sapere cosa marco pensa dei digital e se a massimo piacciono le volvo. Teniamo tutto questo fermo per un momento ed analizziamo un semplice discorso umano:

Andrea, Biagio, Carlo e Diana mangiano cioccolata
Ernesto, Francesco e Giovanna non mangiano cioccolata
Italo, Luana e Marcello vanno in palestra
Nora, Olga e Pasquale bevono birra
la birra è dannosa
la cioccolata è dannosa
la palestra fa bene alla salute

Questo è un tipico ragionamento umano da cui un uomo saprebbe trarre conclusioni anche non esattamente dichiarate nel discorso, vale a dire, ad esempio, che Luana e Marcello tengono alla loro salute e, probabilmente, oltre che andare in palestra, non mangiano nemmeno cioccolata né bevono birra. Il tutto può essere pertanto rappresentato come in figura b. Bene, a costo di ripeterci, la figura b è la vera e propria fase dichiarativa dei domini. Siamo cioè alla dichiarazione:

```
Domains
        persone,cose=symbol
```

Dobbiamo, adesso, sempre nel caso dell'esempio di linguaggio umano appena espresso, passare alle relazioni. Esse possono essere così riassunte, rispettivamente in linguaggio umano:

le persone mangiano le cose
le persone non mangiano le cose
le persone vanno nelle cose
le persone bevono cose
le cose sono cose
le cose fanno cose.

Siamo alla fase della definizione dei predicati: possiamo perciò scrivere:

```
Predicates
mangiare (persone,cose).
non_mangiare (persone,cose).
andare (persone,cose).
bere (persone,cose).
essere (cose,cose)
fare (cose,cose)
```

Ovviamente niente impedisce di usare altri tempi invece degli infiniti per definire l'azione del predicato: ad esempio una dichiarazione di predicato del tipo

```
fa (cosa,cosa)
```

in stretta analogia alla frase di linguaggio naturale «La palestra fa bene alla salute», va lo stesso bene, ma cosa succede se vogliamo esprimere lo stesso concetto utilizzando più cose o desideriamo esprimere un concetto con un verbo passato? Occorrerebbe definire diversi predicati, tutti eseguenti in fondo la stessa operazione, e la cosa diverrebbe impratica; meglio pertanto scegliere solo l'infinito, tanto il Prolog non è ancora riuscito a comprendere la differenza di tempi.

È venuto adesso il momento di definire le clausole, i veri e propri fatti che si desiderano rappresentare. La tecnica più semplice, ma non la più breve, per definire quanto si era detto in parole umane è la seguente:

```
Clauses
mangiare (andrea,cioccolata).
mangiare (biagio,cioccolata).
mangiare (carlo,cioccolata).
mangiare (diana,cioccolata).
non_mangiare (ernesto,cioccolata).
non_mangiare (francesco,cioccolata).
non_mangiare (giovanna,cioccolata).
andare (italo,palestra).
andare (luana,palestra).
andare (marcello,palestra).
bere (nora,birra).
bere (olga,birra).
bere (pasquale,birra).
essere_dannosa (birra).
essere_dannosa (cioccolata).
fare_bene (palestra, salute).
```

È tutto; abbiamo costruito, in Prolog, la nostra base di conoscenza; anche se ci ha comportato una serie abbastanza lunga di battiture alla tastiera; come abbiamo già detto esiste un metodo più semplice per risolvere elegantemente la cosa, ma non è questo il momento di parlarne. Vedremo che cosa fare di tutta questa filastrocca la prossima volta.

MC

POSTAL COMPUTER

OFFERTA SPECIALE: RAM

PC XT IBM COMPATIBILE
L. 750.000
SCHEDA MADRE 6/10 MHZ, 1 DRIVE 360K, SCHEDA CGA O HERCULUS, 256K ESPANDIBILE A 640K SU PIASTRA, TASTIERA AVANZATA 101 TASTI

PC XT IBM COMPATIBILE
L. 1.200.000
SCHEDA MADRE 6/10 MHZ, 1 DRIVE 360K, SCHEDA GRAFICA HERCULUS O CGA, 1 HARD DISK 20 MEGA, 256 ESPANDIBILE A 640K SU PIASTRA, TASTIERA AVANZATA 101 TASTI.

PC AT IBM COMPATIBILE
L. 2.600.000
SCHEDA MADRE 80286 12 MHZ, 0 WAIT, 512 K ESPANDIBILE A 1024 K, 1 DRIVE 5,25'' DA 1.2 MBYTE 1 WINCHESTER DA 40 MBYTE 20MS, SCHEDA SUPER EGA, TASTIERA AVANZATA 101 TASTI.

386 TOWER 16/20 MHZ
L. 5.750.000
MICROPROCESSORE 80286 16/20 MHZ 0 WAIT RAM 2 MB (80 NS) ESPANDIBILE A 16 MB, 8 SLOT, SCHEDA EGA, 1 DRIVE DA 1,2 MB 1 DRIVE 3,5 720 KB, WINCHESTER DA 40 MB.

PREZZI SU RICHIESTA

COVERTASTIERA 84 TASTI	15.000
COVERTASTIERA 101 TASTI	20.000

TUTTI I NS. PREZZI SONO IVA 18%, SPESE SPEDIZIONE ESCLUSA

*** CASSETTE VHS MASTER ***
- HG E 120 L. 4.600 - HG E 180 L. 5.450

TELEFAX MURATA M-1
L. 1.500.000
- COMPATIBILITÀ: G2 G3
- VELOCITÀ DI TRASMISSIONE 15 SECONDI
- APPARECCHIO TELEFONICO A TASTIERA INCORPORATO
- FOTOCOPIATORE
- RICEZIONE AUTOMATICA
- ROTOLO CARTA TERMICA 216 mm × 30 metri.
- OROLOGIO/CALENDARIO DIGITALE

GARANZIA 18 MESI

HARD DISK SEAGATE 20 MB	L. 350.000
HARD DISK CONTROLDATA 40 MB	L. 680.000
HARD DISK CONTROLLER PER XT	L. 100.000
HARD DISK CONTROLLER PER AT	L. 220.000
SCHEDA GRAFICA SUPER E.G.A.	L. 300.000
SCHEDA MULTI I/0	L. 110.000
SCHEDA SERIALE	L. 40.000
SCHEDA PARALLELA	L. 35.000
SCHEDA PORTA JOYSTICK	L. 28.000
SCHEDA MADRE XT	L. 190.000
SCHEDA MADRE AT (12 MHZ 0 WAIT)	L. 650.000
TASTIERA AVANZATA 101 TASTI	L. 110.000
DRIVE 5,25 360KB	L. 140.000
DRIVE 5,25 1,2MB	L. 190.000
DRIVE 3,50 720KB	L. 190.000
DRIVE CONTROLLER	L. 49.000
CAVO PARALLELO	L. 15.000
DATA SWITCH A 2 PORTE	L. 60.000
MOUSE ANKO	L. 59.000
JOYSTICK I.B.M. ANKO	L. 45.000

STAMPANTI CITIZEN GRAFICA - NLQ

CITIZEN 120 D L. 360.000 120 CPS, SET. EPSON IBM 80 COL. TRATO IN TRAZIONE, FRIZIONE INTER. OPZIONALE IBM/COMMODORE	**CITIZEN MSP 50** L. 1050.000 250/300 CAR/SEC., 80 COL.
CITIZEN LSP 100 L. 550.000 -160 cps, 80 COL.	**CITIZEN MSP 55** L. 1.230.000 250/300 CAR/SEC, 136 COL.
CITIZEN MSP 10E L. 650.000 - 160 CAR/SEC, 80 COL.	**CITIZEN HQP 40** L. 1.160.000 - 24 AGHI, 200 CPS ALTISSIMA QUALITÀ
CITIZEN MSP 15E L. 680.000 160 CAR/SEC, 136 COL.	**CITIZEN HQP 45** L. 1.530.000 - 24 AGHI, 200 CPS ALTISSIMA QUALITÀ
CITIZEN MSP 40 L. 775.000 - 200/240 CAR/SEC, 136 COL.	**CITIZEN PREMIERE 35** L. 1.250.000 - MARGHERITA PROFESSIONALE, 35 CPS
CITIZEN MSP 45 L. 950.000 - 200/240 CAR/SEC, 136 COL.	**CITIZEN OVERTURE 110** * L. 3.600.000 - STAMPANTE LASER

TUTTI I PRODOTTI CITIZIEN SONO COPERTI DA CERTIFICATO DI GARANZIA DELLA VALIDITÀ DI DUE ANNI

MONITOR 12'' TTL L. 150.000	F/V
MONITOR 12'' COMPOSITO L. 150.000	Ambra
MONITOR DUAL 12'' L. 200.000	F/V
MONITOR A COLORE MULTITECH L. 555.000	colore ambra F/V
MONITOR PHILIPS COL. 8833 L. 500.000	colore

PORTA FLOPPY 50/60 3 1/2	L. 15.000
PORTA FLOPPY 100 3 1/2	L. 20.000
PORTA FLOPPY 40/50 5 1/4	L. 18.000
PORTA FLOPPY 80/90 5 1/4	L. 23.000

* PRODOTTI COMMODORE SU RICHIESTA *

DISCHETTI
OFFERTA SPECIALE

BULK	10	100	500
5 1/4 DS DD	950	850	750
5 1/4 HD	2200	2100	2000
3 1/2 DS DD	1900	1800	1700

NASHUA	10	100	500
5 1/4 DS DD	1400	1300	1200
5 1/4 HD	2500	2400	2300
3 1/2 DS DD	2200	2000	1900

SU TUTTI I NOSTRI PRODOTTI MAGNETICI OFFRIAMO IL NOSTRO SERVIZIO DI SOSTITUZIONE IMMEDIATA DEI PEZZI DIFETTOSI

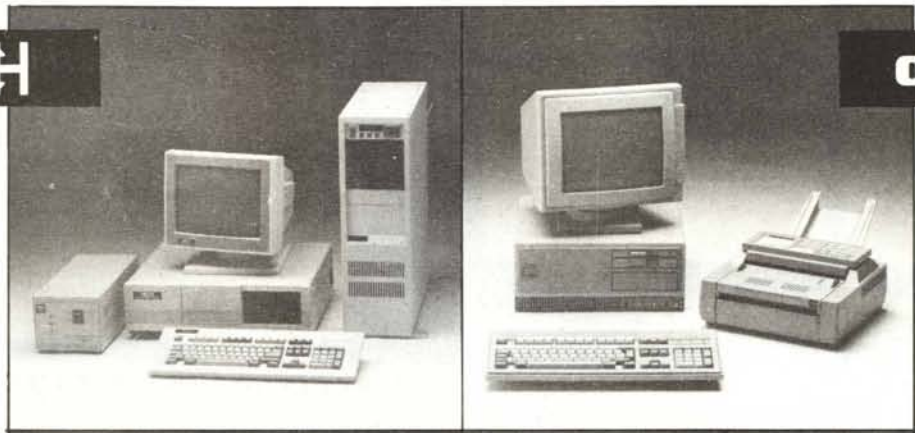
PREZZI IVA 18% ESCLUSA

TEL. 06/3652427/3652431/3650807 TELEFONATECI

SOLO I MIGLIORI.
PER VOI.

HTECH

olivetti



olivetti PRODEST

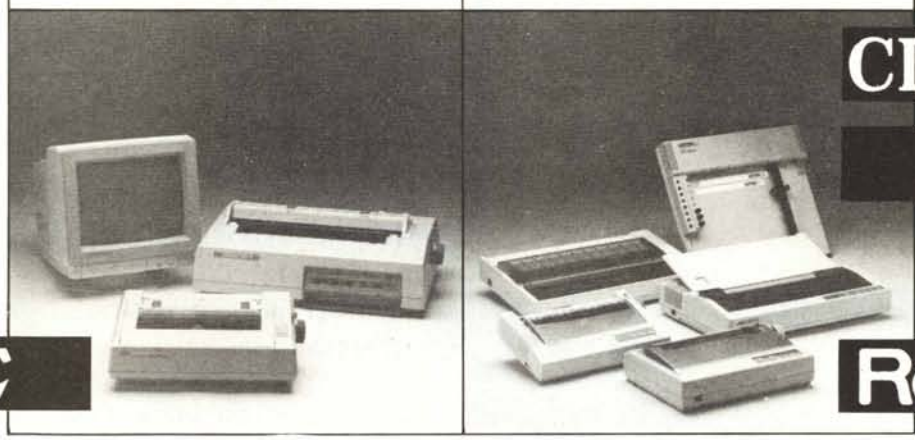
ATARI

CITIZEN

star

NEC

Roland



DISCOM

Discom, ovvero una delle più dinamiche società di distribuzione nate negli ultimi dieci anni. Discom si è imposta sul mercato grazie alla continuità del suo servizio, alla possibilità di offrire il prezzo migliore, alla capacità di scegliere sempre i prodotti vincenti, cioè i migliori, per voi.

00128 Roma - Via Marcello Garosi, 23
Telef. (06) 52.07.839-52.07.917 - Telex 620238 - Telefax (06) 52.05.433