2. STRUCTURA UNUI PROGRAM ÎN PROLOG

Un program în PROLOG este o descriere a unor obiecte şi relaţii existente între aceste obiecte. Obiectele sunt reprezentate prin ***nume simbolice*** a căror structură sintactică este determinată de tipul obiectelor. Există două clase de tipuri, ***tipul elementar*** şi ***tipul complex***. Tipul elementar poate fi ***standard*** sau ***definit de utilizator***. Tipurile definite de utilizator sunt de fapt tot tipuri standard, dar definite prin nume date de utilizator, nume ce sugerează semnificaţia obiectelor de tipul respectiv. Tipul complex la rândul lui se subîmparte în: ***tipul compus*** şi ***tipul listă***. Un nume de tip compus are următoarea structură sintactică: nume simbolic urmat de unul sau mai multe tipuri separate prin virgulă şi închise între paranteze rotunde. Un obiect de tip listă este un şir de obiecte de acelaşi tip, separate prin virgulă şi închise între paranteze drepte. Elementele unei liste pot fi de tip elementar sau de tip compus. Următorul tabel descrie tipurile standard cele mai uzuale. Tipurile ***file***, ***ref***, ***db_selector***, ***bt_selector*** şi ***place*** se referă la fişiere sau baze de date externe şi vor fi prezentate în capitolele referitoare la crearea şi prelucrarea fişierelor şi a bazelor de date externe.

Cuvânt rezervat	Semnificaţie	Exemple
integer	Număr întreg cuprins între -2^{15} şi $2^{15}-1$	1, 0, 456, -17
char	Caracter încadrat între apostrofuri	‘a’, ‘A’, ‘0’, ‘*’
real	Număr real al cărui modul este cuprins între 5.e-324 şi 1.7e+308	5.5, 0.007, 11.897, -8.97
symbol	Şir de caractere în care primul caracter este literă mică	numar, nuMar, numar_1

string	Şir de caractere cuprinse între ghilimele	"numar", "numar", "1abc", "12"

Din punct de vedere sintactic un program în PROLOG se compune din unul sau mai multe module. Unul dintre module este definit ca modul principal. Numai acest modul poate să conțină enunțul unui scop (goal). În general obiectele şi predicatele cu care se lucrează sunt locale fiecărui modul. Dacă un nume de predicat sau tip de obiecte este utilizat în două module, el reprezintă predicate diferite, respectiv tipuri diferite. Există însă şi posibilitatea de defini predicate sau tipuri globale (comune mai multor module). Un modul este alcătuit din mai multe secțiuni. Enumerăm mai jos aceste secțiuni, precizând şi descrierea sintactică a elementelor ce compun fiecare secțiune în parte. Utilizăm convenția de scriere potrivit căreia elementele opționale sunt încadrate între paranteze drepte.

constants

const1 = definiție
const2 = definiție
……

În această secțiune se definesc constantele utilizate în program.

domains

tip_util1 [tip_util2, tip_util3, …] = tip1; tip2;…
lista1 = tip_element*
…….

tip_util1, tip_util2, tip_util3, … , lista1 reprezintă nume simbolice care desemnează tipuri de obiecte definite de programator. tip1; tip2;… reprezintă o listă de tipuri standard sau de tipuri definite de utilizator. Obiectele de tip lista1 sunt liste ale căror elemente sunt de tip tip_element.

global domains

În această secțiune programatorul defineşte tipurile globale. Structura acestei secțiuni este aceeaşi cu cea a secțiunii domains.

database [-nume]

global database [-nume]

În aceste secțiuni se definesc baze de cunoştințe dinamice. Structura lor va fi detaliată în capitolul ce tratează crearea şi utilizarea bazelor de date dinamice.

predicates

predicat1(tip_arg$_1$, tip_arg$_2$, …., tip_arg$_n$)

predicat2(tip_arg$_1$, tip_arg$_2$, …., tip_arg$_k$)

……

Se definesc predicatele local modului. Pentru fiecare predicat se specifică numele şi tipul argumentelor sale.

global predicates

predicat1(tip_arg$_1$, tip_arg$_2$, …., tip_arg$_n$) – (i, o, …) [language C | pascal |…]

predicat2(tip_arg$_1$, tip_arg$_2$, …., tip_arg$_k$)

……..

În această secțiune se precizează predicatele globale şi tipul argumentelor lor, precum şi faptul că anumite argumente sunt de intrare şi altele de ieşire, şi eventual, limbajul în care este scrisă procedura ce corespunde predicatului.

clauses

predicat(arg$_1$, arg$_2$, …, arg$_n$).

predicat$_{k+1}$(…) : - predicat$_1$(…), predicat$_2$(…), …., predicat$_k$(…).

……

…….

Această secțiune este destinată definirii clauzelor. Structura sintactică a clauzelor a fost prezentată în capitolul anterior. Toate clauzele care se referă la acelaşi nume de predicat trebuie grupate.

goal

predicat$_1$(…) [, predicat$_2$(…), …., predicat$_k$(…)]

Scopul (“goal”) unui program PROLOG reprezintă o interogare privind faptele şi regulile definite în secțiunea clauses. Interogarea se poate realiza în două moduri: în secțiunea ***goal*** a programului sau în fereastra ***dialog***. Oricare formă din acestea o exclude pe cealaltă. Structura sintactică a unei interogări este aceeaşi cu structura sintactică a corpului unei reguli, structură prezentată în capitolul anterior.

- Nu toate secțiunile sunt obligatorii.
- Ordinea în care aceste trebuie să apară este următoarea: directive de compilare, domains, global domains, databas, predicates, global predicates, clauses.
- Aşezând directiva ***trace*** în fața secțiunii domains, se poate controla pas cu pas execuția unui scop.
- Secțiunea goal, dacă apare, se aşează fie înainte fie după secțiunea clauses.
- Comentariile pot fi incluse oriunde în program. Un comentariu este un text care începe cu /* şi se termină cu */.
- Scrierea unui program în PROLOG este liberă (se pot scrie mai multe clauze pe un rând, sau o regulă pe mai multe rânduri, etc.). Sintaxa cere ca fiecare clauză să se încheie cu punct.
- În cazul în care se utilizează definiții globale, este de preferat ca acestea să se definească într-un fişier separat, care apoi să fie inclus în fiecare modul cu ajutorul directivei ***include*** care are forma:

include “nume_fişier”