ΕΡΓΑΣΤΗΡΙΟ IX
ΔΥΝΑΜΙΚΗ ΤΡΟΠΟΠΟΙΗΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

ΜΕΡΟΣ Ι

Δυναμική Τροποποίηση Προγράμματος

Η τροποποίηση ενός προγράμματος τη στιγμή που αυτό εκτελείται είναι ένα επικίνδυνο εγχείρημα το οποίο όμως προσφέρει σημαντικές δυνατότητες στον προγραμματιστή της Prolog. Ένα πρόγραμμα που τροποποιεί τον εαυτό του μπορούμε να πούμε ότι είναι ένα πρόγραμμα που "μαθαίνει" από την ίδια την λειτουργία του. Η δυνατότητα της τροποποίησης του προγράμματος την ώρα που αυτό εκτελείται προσφέρεται στην Prolog μέσω κάποιων ενσωματωμένων διαδικασιών.

Τα γεγονότα και οι κανόνες ενός προγράμματος Prolog αποθηκεύονται στη μνήμη του συστήματος. Συνήθως αυτά τα γεγονότα και οι κανόνες "φορτώνονται" στην μνήμη της Prolog από κάποιο αρχείο κειμένου. Υπάρχει όμως και η δυνατότητα ***τροποποίησης των περιεχομένων της μνήμης της Prolog***, με άλλα λόγια του προγράμματος, κατά την διάρκεια εκτέλεσης του. Η προσθήκη/αφαίρεση προτάσεων στην περίπτωση αυτή γίνεται μέσω τριών κατηγορημάτων συστήματος. Με αυτά τα κατηγορήματα μπορούμε να χειριστούμε γεγονότα ή κανόνες του προγράμματος σαν όρους, να ενσωματώσουμε δυναμικά στο πρόγραμμα νέες προτάσεις, ή να διαγράψουμε από το πρόγραμμα γεγονότα ή κανόνες.

Στην κατηγορία αυτή ανήκουν οι ακόλουθες ενσωματωμένες διαδικασίες:

- ☑ **asserta(X)** : Το γεγονός ή κανόνας Χ ενσωματώνεται στο πρόγραμμα πριν από τις προτάσεις του προγράμματος με το ίδιο κατηγόρημα για κεφαλή.
- ☑ **assert(X)**: Το γεγονός ή κανόνας Χ ενσωματώνεται στο πρόγραμμα κάτω από τις προτάσεις του προγράμματος με το ίδιο κατηγόρημα για κεφαλή.
- ☑ **retract(X)** : Το πρώτο γεγονός ή κανόνας του προγράμματος που η κεφαλή του ταυτοποιείται με το κατηγόρημα Χ αφαιρείται από την μνήμη.

Οι προτάσεις που εισάγονται στο πρόγραμμα με τη βοήθεια των διαδικασιών αυτών συμπεριφέρονται με τον ίδιο ακριβώς τρόπο όπως και όλες οι άλλες προτάσεις του προγράμματος.

Όταν τα κατηγορήματα που θέλουμε να προσθέσουμε ή να αφαιρέσουμε, χρησιμοποιώντας τις παραπάνω διαδικασίες δεν παρουσιάζονται σε άλλο σημείο του προγράμματος, αλλά μόνο μέσα σε αυτές τις διαδικασίες δεν χρειάζεται να κάνουμε κάποια επιπλέον δήλωση στο πρόγραμμα. Αν όμως τα ίδια κατηγορήματα χρησιμοποιούνται και σε άλλες προτάσεις μέσα στο πρόγραμμα τότε απαιτείται δήλωση του τύπου :

:- dynamic κατηγόρημα /αριθμός ορισμάτων.

η οποία δηλώνει ποιες διαδικασίες είναι δυνατόν να τροποποιηθούν κατά τη διάρκεια εκτέλεσης του προγράμματος μέσω των κατηγορημάτων assert και retract.

Πρόγραμμα : lab9_1.pl

```
:-dynamic factorial/2.
factorial(0,1).
factorial(N,F):- N1 is N-1,
        factorial(N1,F1),
        F is N*F1,
        asserta(factorial(N,F)).
```

Πρόγραμμα : lab9_2.pl

```
factorial(0,1).
factorial(N,F):- N1 is N-1,
        factorial(N1,F1),
        F is N*F1.
```

Προσομοίωση Καθολικών Μεταβλητών

Χρησιμοποιώντας τα κατηγορήματα συστήματος assert / retract μπορούμε να αποθηκεύσουμε την τιμή μιας μεταβλητής για να χρησιμοποιηθεί από όλες τις προτάσεις του προγράμματος.

Έστω ότι έχουμε τα εξής γεγονότα (lab9_3.pl):

next("Hellas","Turkey").
next("Hellas ","Albania").
next("Hellas ","Bulgaria").
next("Hellas ","FYROM").
next("Turkey","Iraq").
…

και θέλουμε να φτιάξουμε ένα πρόγραμμα το οποίο θα βρίσκει με πόσες και ποιες χώρες συνορεύει μια χώρα.

```
count(Country,T) :-  assert(counter(0)), fail.
count(Country,T) :-  next(Country,X), name(Y,X), write(Y), nl,
           retract(counter(T)), T1 is T+1,
     assert(counter(T1)), fail.
count(Country,T) :-  retract(counter(T)), write(T), nl.
```

Σημείωση : Η διαδικασία **name(X,Y)** μετατρέπει ένα συμβολικό όνομα X στη λίστα Y των ASCII κωδικών των χαρακτήρων που αποτελούν το όνομα και αντίστροφα.

Χειρισμός συνόλου λύσεων

Μια άλλη σημαντική ανάγκη για τον προγραμματισμό στην Prolog είναι η δυνατότητα συλλογής όλων των εναλλακτικών λύσεων μιας ερώτησης σε μια λίστα ώστε να είναι εύκολη η παραπέρα επεξεργασία τους σαν σύνολο. Η δυνατότητα αυτή προσφέρεται μέσω μιας ομάδας ενσωματωμένων διαδικασιών :

- **bagof(X,K,L)** : Προκύπτει η λίστα L η οποία περιλαμβάνει όλα τα X για τα οποία αληθεύει ο ατομικός τύπος K.

- **setof(X,K,L)** : Είναι παρόμοιο με το προηγούμενο, με τη διαφορά ότι έχουν αφαιρεθεί τα διπλά X που ενδεχόμενα περιλάμβανε η λίστα L.

- **findall(X,K,L)** : Προκύπτει η λίστα L η οποία περιλαμβάνει όλα τα X για τα οποία αληθεύει ο ατομικός τύπος K. Η διαφορά του με το bagof είναι ότι στη γενική περίπτωση (χρήση μεταβλητών) επιστρέφει σε μια λίστα όλα τα αποτελέσματα που αφορούν τον ατομικό τύπο K.

ΜΕΡΟΣ ΙΙ

Βοηθητικά προγράμματα (Βρίσκονται στην Ασύγχρονη Εκπαίδευση)

Άσκηση Α	exercise9_1.pl exercise9_2.pl
Άσκηση Β	exercise9_3.pl
Άσκηση Γ	exercise9_4.pl

ΑΣΚΗΣΗ Α

Διατυπώστε τις ερωτήσεις factorial(4,X) και factorial(5,X) για κάθε μια από τις δυο υλοποιήσεις του κατηγορήματος factorial/2. Παρατηρείστε την αναζήτηση που κάνουν για να δώσουν απάντηση (Προσπαθήστε να δείτε πόσες φορές εκτελείται η αναδρομή σε κάθε πρόγραμμα).

ΑΣΚΗΣΗ Β

i) Διατυπώστε την ερώτηση

```
?- count("Hellas",T).
```

και ελέγξτε την αναζήτηση που κάνει η Prolog για να δώσει την απάντηση.
Τι θα γινόταν αν δεν χρησιμοποιούσαμε το κατηγόρημα name(X,Y);

ii) Τροποποιήστε το πρόγραμμα έτσι ώστε να εκτυπώνει την τελευταία γραμμή με την εξής μορφή:

«Η χώρα X συνορεύει με Y χώρες»

όπου X το όνομα της χώρας που εισήχθη (πχ Hellas) και Y ο αριθμός των γειτόνων της (πχ 4).

ΑΣΚΗΣΗ Γ

Δίνεται το πρόγραμμα:

```
likes(driving,maria).
likes(swimming,maria).
likes(swimming,lari).
likes(painting,eva).
likes(writing,george).
likes(driving,john).
likes(swimming,lari).
```

Όσοι ανήκουν στη λίστα List είναι άτομα τα οποία έχουν κάποιο χόμπι.

```
have_hobbies_set(X,List):-setof(Y, likes(X,Y),List).
have_hobbies_bag(X,List):-bagof(Y, likes(X,Y),List).
have_hobbies_all(X,List):- findall(Y, likes(X,Y),List).
```

Υπολογισμός του πλήθους N των ατόμων που έχουν το χόμπι X:

```
no_of_people(Y,N):- have_hobbies_set(Y,List), length(List, N).
```

Ποιά ερώτηση πρέπει να κάνετε για να βρείτε μία λίστα που θα περιέχει <u>όλα τα άτομα</u> που έχουν κάποιο χόμπι;