ΠΛΗ31

PROLOG

Μάθημα 3:
Δομές και Λίστες

Δημήτρης Ψούνης

ΠΕΡΙΕΧΟΜΕΝΑ

Α. Σκοπός του Μαθήματος

Β.Θεωρία

1. Δομές
    1. Ορισμός Δομής
    2. Ερωτήσεις σε Δομές
2. Λίστες
    1. Ορισμός Λίστας
    2. Κατηγορήματα Λιστών
        1. Το κατηγόρημα member/2
        2. Το κατηγόρημα append/3
        3. Το κατηγόρημα length/2
        4. Το κατηγόρημα reverse/3

Γ.Ασκήσεις

Α. Θεωρία

1. Δομές

1.Ορισμός Δομής

- Στην Prolog μπορούμε να ορίσουμε μια δομή δεδομένων ως μία ομαδοποίηση δεδομένων με ένα κοινό όνομα και αντίστοιχο συντακτικό με αυτό που είδαμε στο κατηγόρημα.
  - Π.χ. το date(18,12,2011) θα αναπαριστά την ημερομηνία 18-12-2011

- Μία δομή δεδομένων παίζει αντίστοιχο ρόλο με μία σταθερά, άρα μπορούμε να την αναπαράστήσουμε μόνο σαν όρισμα σε ένα κατηγόρημα.

- Για παράδειγμα μπορούμε να ενσωματώσουμε στον κόσμο της οικογένειας του tom και την πληροφορία για την ημερομηνία γέννησης κάθε προσώπου με τις δηλώσεις:

```
born(tom,date(12,11,1923)).
born(john,date(17,5,1955)).
born(pam,date(10,11,1978)).
```
κ.λπ.

Α. Θεωρία

1. Δομές

2. Ερωτήσεις σε Δομές

- Έπειτα μπορούμε να χρησιμοποιήσουμε την πληροφορία των δομών με τους γνωστούς τρόπους απάντησης της Prolog:

```
?- born(tom,X).
X=date(12,11,1923).

?-born(tom,date(_,_,X)).
X=1923

?-parent(X,john),born(X,date(_,_,Y)).
X=tom,
Y=1923.
```

Α. Θεωρία

2. Λίστες

1. Ορισμός Λίστας

- Η λίστα είναι μια βασική προγραμματιστική δομή στην Prolog, η οποία είναι να ενθέσουμε τα στοιχεία της λίστας σε αγκύλες χωρισμένα με κόμματα.
  - Η λίστα είναι μια ιδιαίτερα σημαντική δομή διότι μπορούμε να έχουμε όσα στοιχεία θέλουμε, σε αντίθεση με ένα κατηγόρημα που έχει συγκεκριμένο πλήθος στοιχείων,
- Συγκεκριμένα αν θέλουμε να κατασκευάσουμε μια λίστα π.χ. με τα παιδιά ενός ατόμου, τότε μπορούμε να το γράψουμε απ' ευθείας ως εξής:

```
children(tom,[john,jim]).
```

- Στην πραγματικότητα η λίστα κωδικοποιείται στην Prolog με έναν ιδιαίτερο τρόπο, χρησιμοποιώντας τα δεσμευμένα κατηγορήματα ./2 και [ ]

```
children(tom,.(john,.(jim,[ ]))).
```

  - Αλλά ευτυχώς παρέχεται και η προγραμματιστική ευκολία να γράψουμε τα στοιχεία απ' ευθείας σε άγκύλες

Α. Θεωρία

2. Λίστες

1. Ορισμός Λίστας

- Με βάση και τον τυπικό τρόπο με τον οποίο η Prolog κατασκευάζει μια λίστα, μας παρέχεται μια πολύ σημαντική προγραμματιστική ευκολία, που θα την χρησιμοποιήσουμε όταν κάνουμε ερωτήσεις και όταν γράφουμε κανόνες:
  - Μία λίστα μπορούμε να αναπαρασταθεί [Head|Tail] όπου
    - Head είναι η κεφαλή της λίστας, δηλαδή το πρώτο στοιχείο της λίστας
    - Tail είναι η ουρά της λίστας, δηλαδή τα επόμενα στοιχεία της λίστας.
  - Έτσι αν π.χ. έχουμε ενσωματώσει στο πρόγραμμα μας το γεγονός:

```
children(a,[b,c,d,e]).
```

  - Και κάνουμε την ερώτηση:

```
?-children(a,[X|Y]).
```

  - Θα λάβουμε την απάντηση

```
X=b,
Y=[c,d,e].
```

Α. Θεωρία

2. Λίστες

2. Κατηγορήματα Λιστών (1.Το κατηγορημα member/2)

- Η Prolog έχει μια σειρά έτοιμων κατηγορημάτων για την διαχείριση λιστών.
  - Τα κατηγορήματα αυτά είναι έτοιμα και δεν χρειάζεται να τα ορίσουμε εμείς. Απλά τα χρησιμοποιούμε.

- Το σημαντικότερο κατηγόρημα είναι το member(X,L) που θα αληθεύει αν το στοιχείο X είναι μέλος της λίστας L.
- Παραδείγματα:

```
?- member(1,[1,2,3]).
true.

?- member(4,[1,2,3]).
false.

?- member(X,[1,2,3]).
X = 1 ;
X = 2 ;
X = 3.
```

Α. Θεωρία

2. Λίστες

2. Κατηγορήματα Λιστών (1.Το κατηγορημα member/2)

- Ο ορισμός του κατηγορήματος member στην prolog είναι προφανώς αναδρομικός:

```
member(X,[X|_]).
member(X,[_|L]):-member(X,L).
```

- (Ασκηση) Να κατασκευάσετε το δένδρο εκτέλεσης του ερωτήματος

```
?- member(X,[1,2,3]).
```

Α. Θεωρία

2. Λίστες

2. Κατηγορήματα Λιστών (2.Το κατηγορημα append/2)

- Σημαντικό κατηγόρημα είναι και το append(L1,L2,L) που θα αληθεύει αν η συνένωση των λιστών L1 και L2 είναι η L.
  - Συνένωση των λιστών L1 και L2 είναι να κατασκευάσουμε μία νέα λίστα L που στην αρχή έχει τα στοιχεία της L1 και έπειτα έχει και τα στοιχεία της L2.

- Παραδείγματα:

```
10 ?- append([1,2],[3,4],L).
L = [1, 2, 3, 4].
```

```
?- append(L1,[3,4],[1,3,4]).
L1 = [1]
```

```
?- append([1,2],L2,[1,3,4]).
false.
```

```
?- append(L1,L2,[1,2,3]).
L1 = [],
L2 = [1, 2, 3] ;
L1 = [1],
L2 = [2, 3] ;
L1 = [1, 2],
L2 = [3] ;
L1 = [1, 2, 3],
L2 = [] ;
```

Α. Θεωρία

2. Λίστες

2. Κατηγορήματα Λιστών (2.Το κατηγορημα append/2)

- Και ο ορισμός του κατηγορήματος append στην prolog είναι αναδρομικός:

```
append([],L,L).
append([X|L1],L2,[X|L3]):-append(L1,L2,L3).
```

- (Ασκηση) Να κατασκευάσετε το δένδρο εκτέλεσης του ερωτήματος

```
?- append([1,2],[3],L).
```

Α. Θεωρία

2. Λίστες

2. Κατηγορήματα Λιστών (3.Το κατηγορημα length/2)

- Το κατηγόρημα length(L,X) αληθεύει αν η λίστα L έχει μήκος (πλήθος στοιχείων) X
- Παραδείγματα:

```
?- length([1,2,3],X).
X = 3.
```

```
?- append(L1,L2,[1,2,3]),length(L1,N1),length(L2,N2).
L1 = [],
L2 = [1, 2, 3],
N1 = 0,
N2 = 3 ;
L1 = [1],
L2 = [2, 3],
N1 = 1,
N2 = 2 ;
L1 = [1, 2],
L2 = [3],
N1 = 2,
N2 = 1 ;
L1 = [1, 2, 3],
L2 = [],
N1 = 3,
N2 = 0 ;
```

Α. Θεωρία

2. Λίστες

2. Κατηγορήματα Λιστών (4.Το κατηγορημα reverse/2)

- Το κατηγόρημα reverse(L1,L2) αληθεύει αν η λίστα L2 είναι η αντιστροφή της L1
- Παραδείγματα:

```
?- reverse([1,2,3],L).
L = [3, 2, 1].

10 ?- reverse(L,[1,2,3]).
L = [3, 2, 1] .
```

Β. Ασκήσεις

1. Εξετάσεις 2009Β

Το κατηγόρημα append της Prolog ορίζεται ως εξής:

```
append([],L,L).
append([X|Xs],Y,[X|Zs]) :- append(Xs,Y,Zs).
```

Να γράψετε τι θα απαντήσει η Prolog στα ακόλουθα ερωτήματα (όλες τις πιθανές απαντήσεις):

(1) `?- append([1,2],[3,4],X).`
(2) `?- append([a,b],Y,[a,b,c,d]).`
(3) `?- append(Z,[a,b],[a,b,c,d]).`
(4) `?- append(A,B,[k,l,m]).`

Β. Ασκήσεις

2. Ο κόσμος του οικογενειακού δένδρου

Δεδομένου του γνωστού κόσμου του οικογενειακού δένδρου του tom:

```
parent(tom,bob).
parent(tom,john).
parent(bob,jim).
parent(bob,kim).
parent(john,sam).
parent(jim,pat).
```

- Ορίστε το κατηγόρημα children(X,L), που θα αληθεύει αν τα στοιχεία που είναι στην λίστα L είναι παιδιά του ατόμου X.