

Ασκήσεις - Αριθμητικές Διαδικασίες

- Να ορισθεί η σχέση **sign** (πρόσημο) που να επιστρέφει την ένδειξη **positive, negative, zero** ανάλογα με την τιμή της παραμέτρου.

```
sign(X, positive) :-  
    X > 0.  
sign(X, zero) :-  
    X == 0.  
sign(X, negative) :-  
    X < 0.
```

16

Ασκήσεις - Αριθμητικές Διαδικασίες

- Δίνεται μια βάση με γεωγραφικά στοιχεία πληθυσμού (**population**) και έκτασης (**area**) διαφόρων περιοχών. Να ορισθεί η σχέση **density** που να επιστρέφει την πυκνότητα του πληθυσμού, δηλαδή τον αριθμό των κατοίκων μιας περιοχής ανά τετραγωνικό χιλιόμετρο.

```
population(usa,203) .           area(usa,3) .  
population(india,750) .         area(india,1) .  
population(china,1200) .        area(china,4) .  
population(brazil,112) .        area(brazil,3) .
```

```
density(X,Y) :-  
    population(X,P) ,  
    area(X,A) ,  
    Y is P / A.
```

17

Ασκήσεις - Αριθμητικές Διαδικασίες και Αναδρομή

- Να ορισθεί διαδικασία για τον υπολογισμό παραγοντικού

```
fact(0,1) .  
fact(N,F) :-  
    N > 0 ,  
    N1 is N - 1 ,  
    fact(N1,F1) ,  
    F is N * F1
```
- Να ορισθεί η διαδικασία **between(N1,N2,X)**, η οποία να παράγει όλους τους ακραίους στο διάστημα **[N1,N2]**.

```
between(N,N,[N]) .
```

```
between(N1,N2,[N1|T]) :-  
    N1 < N2 ,  
    NewN1 is N1 + 1 ,  
    between(NewN1,N2,T) .
```

18

Ασκήσεις - Αριθμητικές Διαδικασίες και Λίστες

- Να ορισθεί η σχέση **length** που να επιστρέφει το μήκος μιας λίστας στοιχείων

```
length([],0) .  
length([_|T],X) :-  
    length(T,X1) ,  
    X is X1 + 1.
```
- Να ορισθεί διαδικασία για το εσωτερικό γινόμενο 2 διανυσμάτων.

```
list_prod([],[],0) .  
list_prod([X1|T1],[X2|T2],P) :-  
    list_prod(T1,T2,P2) ,  
    P is X1*X2 + P2.
```
- Να γραφεί πρόγραμμα που να υπολογίζει το διανυσματικό άθροισμα 2 διανυσμάτων (διανύσματα = λίστες).

```
addlist([],[],[]) .  
addlist([X|T1],[Y|T2],[Z|T3]) :-  
    Z is X + Y ,  
    addlist(T1,T2,T3) .
```

19

Επαναληπτικές και Αναδρομικές Διαδικασίες

- Να γραφεί διαδικασία, η οποία να υπολογίζει το άθροισμα των στοιχείων μιας λίστας. Υπάρχουν 2 εκδοχές:
 - Καθαρά αναδρομικός ορισμός

```
sum_list([],0).
```

```
sum_list([H|T], Sum) :-
```

```
    sum_list(T, TempSum),
```

```
    Sum is H + TempSum.
```

- "Επαναληπτικός" ορισμός με τη χρήση βοηθητικού κατηγορήματος

- ✓ Η επανάληψη ονομάζεται έτσι γιατί διαδικαστικά θυμίζει την επανάληψη των κλασικών γλωσσών προγραμματισμού

Θέτουμε μία αρχική τιμή

```
sum_list(List,Sum) :-
```

```
    sum_list_aux(List,0,Sum).
```

```
sum_list_aux([],Sum,Sum).
```

```
sum_list_aux([H|T],Temp,Sum) :-
```

```
    Next is Temp + H,
```

```
    sum_list_aux(T,Next,Sum).
```

Σε κάθε επανάληψη υπολογίζουμε την επόμενη προσωρινή τιμή με βάση την προηγούμενη προσωρινή τιμή (ξεκινώντας από την αρχική)

Όταν αληθεύει η συνθήκη τερματισμού, τότε η προσωρινή τιμή είναι και το τελικό αποτέλεσμα

20

Επανάληψη vs. Αναδρομή

- ❖ Οι επαναληπτικές διαδικασίες υλοποιούνται με κανονική αναδρομή της Prolog
 - Δεν εισάγεται καμία καινούργια έννοια, απλά αλλάζει λίγο ο τρόπος σκέψης και προγραμματισμού
 - ✓ Χρησιμοποιούμε βοηθητικό κατηγορήμα και βοηθητικές παραμέτρους, οι οποίες κρατάνε τις προσωρινές τιμές.
- ❖ Πλεονεκτήματα:
 - Η εκτέλεση της αναδρομής καταναλώνει λιγότερο χώρο στη μνήμη, λόγω βελτιστοποίησης ουραίας κλήσης (**tail-recursion optimization**)
 - ✓ Επειδή η αναδρομική κλήση είναι η τελευταία κλήση που υπάρχει στον ορισμό του κατηγορήματος, η Prolog δε χρειάζεται να τοποθετήσει στη στοίβα στοιχεία για τις μεταβλητές καθώς και για τα σημεία επιστροφής και οπισθοδρόμησης
 - Πολλές φορές, προγράμματα που δεν "τρέχουν" λόγω μνήμης, με τέτοιες αλλαγές στον κώδικα είναι σε θέση να τρέξουν
- ❖ Μειονεκτήματα:
 - Ο κώδικας είναι λιγότερο "λογικός" ή "δηλωτικός" και περισσότερο "διαδικαστικός",
 - ✓ Δηλαδή η Prolog θυμίζει τις κλασικές γλώσσες προγραμματισμού!
 - Χρειάζεται παραπάνω κατηγορήματα και ορίσματα
 - Στις διαδικασίες χειρισμού λιστών, υπάρχει περίπτωση οι λύσεις να επιστρέφονται αντεστραμμένες
 - ✓ Χρειάζεται η χρήση του κατηγορήματος **reverse/2**

21

Μαθηματικές Συναρτήσεις

- Υποστηρίζεται ένας αριθμός συναρτήσεων για τον υπολογισμό τριγωνομετρικών, λογαριθμικών ή άλλων μαθηματικών συναρτήσεων, οι οποίες διαφέρουν στις διάφορες εκδόσεις.
- Αναφέρονται ενδεικτικά μερικές :

R is cos(N) R = συν(N)

R is sin(N) R = ημ(N)

R is tan(N) R = εφ(N)

R is exp(N) R = e^N

R is sqrt(N) R = \sqrt{N}

22