

ΑΣΚΗΣΗ ΠΡΑΞΗΣ 2: Υλοποίηση Αναδρομικών Προγραμμάτων στην Prolog

Στα μαθηματικά μια αναδρομική συνάρτηση μπορεί να ορίζεται αναφερόμενη στον εαυτό της, όπως για παράδειγμα η συνάρτηση $\text{factorial}(N)$ (N παραγοντικό) η οποία ως γνωστόν ορίζεται από τον αναδρομικό τύπο:

$$\text{factorial}(N) = \text{factorial}(N-1) * N$$

σε συνδυασμό με την οριακή συνθήκη:

$$\text{factorial}(1) = 1$$

Η οριακή συνθήκη είναι απαραίτητη για να τερματίσει κάποια στιγμή μία αναδρομική διαδικασία εκτέλεσης. Αν υπήρχε μόνο η αναδρομική σχέση χωρίς την οριακή συνθήκη τότε, για παράδειγμα, η τιμή $\text{factorial}(3)$ θα αναφερόταν στο $\text{factorial}(2)$, το οποίο θα αναφερόταν στο $\text{factorial}(1)$, το οποίο θα αναφερόταν στο $\text{factorial}(0)$, το οποίο θα αναφερόταν στο $\text{factorial}(-1)$, κλπ, μια διαδικασία δηλαδή που προφανώς θα συνεχιζόταν επ' άπειρον.

Πώς υλοποιούμε μαθηματικές συναρτήσεις (ή και άλλα προβλήματα) με τη μέθοδο της αναδρομής; Ας θυμηθούμε την αρχή της επαγωγής που εισήγαγε ο Αριστοτέλης για την απόδειξη θεωρημάτων: «έστω ότι έχουμε λύσει το πρόβλημα για $N-1$ οπότε ψάχνουμε τη λύση για N ». Η μέθοδος της αναδρομής είναι αντίστροφη της επαγωγής. (Η επαγωγή στην ουσία χρησιμοποιείται για την απόδειξη μίας αναδρομικής σχέσης). Ποιο συγκεκριμένα:

Μέθοδος αναδρομής για τις συναρτήσεις:

Έστω ότι θέλω να βρω την τιμή μιας συνάρτησης $f(x)$ για $x=N$. Θεωρώ ότι έχω βρει την τιμή της f για $x=N-1$ και ψάχνω να βρω τη σχέση μεταξύ του $f(N)$ και του $f(N-1)$.

Μία αναδρομική σχέση πρέπει να συνοδεύεται πάντα από μια τουλάχιστον οριακή περίπτωση (που παίζει το ρόλο της τερματικής συνθήκης) για κάποια οριακή τιμή του N , η οποία είναι συνήθως (αλλά όχι πάντα!) $N=0$ ή $N=1$.

Υλοποίηση αναδρομικών συναρτήσεων στην Prolog

Καθώς η αναδρομή είναι ουσιαστικά εκείνο το υπολογιστικό εργαλείο που κυριαρχεί στην Prolog η υλοποίηση αναδρομικών συναρτήσεων σ' αυτή τη γλώσσα είναι εύκολη και σχεδόν αυτόματη, αν γνωρίζουμε το μαθηματικό αναδρομικό τύπο της συνάρτησης. Η μόνη λεπτομέρεια που πρέπει να προσεχθεί είναι ότι στην Prolog δεν υπάρχουν υποπρογράμματα-συναρτήσεις αλλά κατηγορήματα, και άρα η έξοδος y μιας συνάρτησης $f(x)$ πρέπει να τοποθετηθεί σαν ένα επί πλέον όρισμα στο κατηγορημα $\text{fp}(X, Y)$ που υλοποιεί τη συνάρτηση αυτή στην Prolog.

Για παράδειγμα, το κατηγορημα $\text{fact}(N, Y)$ που ορίζεται παρακάτω υλοποιεί τη συνάρτηση $Y = \text{factorial}(N)$.

```
fact(1, 1).
fact(N, Y) :-
    N > 1,
    N1 is N-1,
    fact(N1, Y1),
    Y is Y1*N.
```

Βασική Περίπτωση
(ή Οριακή συνθήκη)

Αναδρομικός κανόνας



Παρατήρηση1: Το κατηγορημα **is** είναι ενσωματωμένο στην Prolog (κατηγορημα βιβλιοθήκης) και χρησιμοποιείται όταν θέλουμε να υπολογίσουμε την τιμή κάποιας μαθηματικής έκφρασης. Υπενθυμίζουμε ότι η Prolog ως συμβολική γλώσσα προγραμματισμού δεν υπολογίζει - **σκόπιμα** - τις εκφράσεις που εμφανίζονται στον κώδικά της. Όταν χρησιμοποιούμε το κατηγορημα **is** η Prolog χάνει τη δυνατότητα της αντίστροφης χρήσης των ορισμάτων του κατηγορηματος. Στη συγκεκριμένη περίπτωση μπορούμε μόνο να ρωτήσουμε **?- fact(3, A).** και όχι **?- fact(N, 6).**

Παρατήρηση2: Η τοποθέτηση της συνθήκης $N > 1$ στον παραπάνω κανόνα καθιστά τις δύο φράσεις αμοιβαία αποκλυόμενες και διασφαλίζει το γεγονός ότι η συνάρτηση factorial ορίζεται μόνο για $N \geq 0$.

Βοηθητικά κατηγορήματα για την άσκηση:

- **read(X): Ανάγνωση από το πληκτρολόγιο**

Το ενσωματωμένο κατηγορήμα **read(X)** παίρνει ένα μόνο όρισμα X , το οποίο μπορεί να είναι ένας οποιοσδήποτε όρος (term) της Prolog. Η λειτουργία του read έχει ως εξής: η Prolog περιμένει να γράψουμε κάτι στο πληκτρολόγιο, π.χ. ένα string όπως 'Hello', έναν αριθμό όπως 2435, ή ένα συμβολικό όνομα όπως john, ή ένα σύνθετο όρο όπως name(nikos,paradakis) και το στοιχείο αυτό καταχωρείται στο X , δηλαδή η μεταβλητή X δεσμεύεται στην τιμή που πληκτρολογήσαμε.

Προσοχή! Το read, στη διαδικασία εκτέλεσης, περιμένει μια τελεία (.) στο τέλος του input και στη συνέχεια <enter> για να σταματήσει να διαβάζει. Αν ξεχάσουμε να πληκτρολογήσουμε την τελεία η read παραμένει σε κατάσταση αναμονής και δίνει την εντύπωση ότι το σύστημα κόλλησε.

- **write(X): Εμφάνιση/εκτύπωση πληροφορίας στην οθόνη.**

Το ενσωματωμένο κατηγορήμα **write(X)** παίρνει επίσης ένα μόνο όρισμα X , το περιεχόμενο του οποίου εμφανίζει στην οθόνη.

- **nl: Αλλαγή γραμμής**

Χρησιμοποιείται μετά από μία **write** για αλλαγή γραμμής στην οθόνη όπου εμφανίζονται τα δεδομένα.

ΤΙ ΠΡΕΠΕΙ ΝΑ ΚΑΝΕΤΕ!

(α) Ένας θετικός αριθμός X είναι φυσικός αν ο προηγούμενός του $X-1$ είναι φυσικός. Ο αριθμός 0 είναι φυσικός. Να γραφεί ο αναδρομικός ορισμός Prolog ενός κατηγορήματος το οποίο δεδομένου ενός αριθμού, να βρίσκει αν αυτός ο αριθμός είναι φυσικός. Για παράδειγμα:

```
?- natural(21).
yes
?- natural(2.5).
No
```

(β) Γράψτε το κατηγορήμα **power(X, N, P)** το οποίο υλοποιεί το ύψωμα σε δύναμη: $P=X^N$. Για παράδειγμα:

```
?- power(3, 5, X).
X = 243

?- power(4, 3, X).
X = 64

?- power(2, 4, X).
X = 16
```

(γ) Γράψτε το κατηγορήμα **fibonacci(N, Y)** το οποίο υλοποιεί τη συνάρτηση Fibonacci:

```
fibonacci(N) = fibonacci(N-1) + fibonacci(N-2)
fibonacci(2) = 1
fibonacci(1) = 1
```

Προσέξτε ότι η **fibonacci(N)** έχει δύο οριακές συνθήκες (ο λόγος είναι ότι ο αναδρομικός τύπος έχει δύο αναφορές στη **fibonacci()**, μια για N και μια για $N-1$). Παραδείγματα:

```
?- fibo(3, X).
X = 2

?- fibo(4, X).
X = 3

?- fibo(5, X).
X = 5

?- fibo(6, X).
X = 8
```

N	1	2	3	4	5	6	7	8
Fib	1	1	2	3	5	8	13	

```
?- fibo(8, X).
X=
```

(δ) Γράψτε το κατηγορήμα `mkd(N, M, D)` το οποίο υλοποιεί τη συνάρτηση του Μέγιστου Κοινού Διαιρέτη (ΜΚΔ) μεταξύ των αριθμών N και M :

$$\text{MKΔ}(N, M) = \text{MKΔ}(M, N), \text{ αν } N < M$$

$$\text{MKΔ}(N, M) = \text{MKΔ}(M, \text{mod}(N, M)), \text{ αν } N \geq M$$

$$\text{MKΔ}(N, 0) = N$$

Η συνάρτηση `mod(N, M)` είναι το γνωστό modulo δηλαδή το υπόλοιπο της ακεράιας διαίρεσης μεταξύ N και M . Η συνάρτηση `mod(N, M)` είναι από τις λίγες συναρτήσεις που υπάρχει αυτούσια στην Prolog. Παραδείγματα:

?- `mkd(3, 6, X)` .

`X = 3`

?- `mkd(10, 4, X)` .

`X = 2`

?- `mkd(7, 12, X)` .

`X = 1`

?- `mkd(24, 60, X)` .

`X = 12`

A/A	N	M
1	8	22
2	22	8
3	8	6
4	6	2
5	2	0

?- `mkd(8, 22, X)` .

`X =`

(ε) Γράψτε ένα κατηγορήμα `run` χωρίς ορίσματα το οποίο θα ζητά από το χρήστη να εισάγει 2 αριθμούς A , B , και θα τυπώνει στην οθόνη τα εξής αποτελέσματα: `AB`, `fibonacci(A)`, και `mkd(A,B)`. Παράδειγμα:

?- `run`.

`Dwse ton arithmo A:`

`6.`

`Dwse ton arithmo B:`

`4.`

`H dynamh A^B einai 1296`

`fibonacci(A) = 8`

`O megistos koinos diaireths A, B einai 2`

Βοήθημα:

`write('Hello')` → Hello

`nl` → αλλαγή γραμμής εκτύπωσης

π.χ.

?-`X is 6*2, write('Hi'),nl, write('The sum is '), write(X).`

Hi

The sum is 12