Αλεξάνδρειο ΤΕΙ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
Τμήμα Μηχανικών Πληροφορικής ΑΤΕΙΘ

Αντικειμενοστρεφής Προγραμματισμός

Εργαστήριο 1

Αναδρομή (Recursion)

1. Εισαγωγή

Η αναδρομή (recursion) είναι μία διαδικασία ανασκόπησης και μία προγραμματιστική τεχνική. Οι μέθοδοι (υποπρογράμματα, συναρτήσεις, διαδικασίες) μπορεί να είναι *αναδρομικές (recursive)* δηλ. να καλούν τον εαυτό τους.

Τα παραδείγματα που χρησιμοποιούνται ως αναδρομικά υποπρογράμματα είναι κάποιες μαθηματικές συναρτήσεις, αλλά και κάποιοι πιο πολύπλοκοι αλγόριθμοι και δομές.

Παράδειγμα αναδρομικού ορισμού:
Όρισμός: Ο απόγονος ενός ατόμου είναι:
ένα παιδί του ή
ο απόγονος ενός παιδιού του.

Ο ορισμός είναι αναδρομικός γιατί περιγράφεται χρησιμοποιώντας τον εαυτό του.

Σημαντικά σημεία:

- Κάθε αναδρομικός αλγόριθμος (recursive algorithm) πρέπει να έχει ένα σημείο τερματισμού του. Στον υπολογισμό του παραγοντικού είναι όταν **n** = 0. Σε όλους τους αναδρομικούς αλγόριθμους θα πρέπει να υπάρχει κάτι το οποίο κάποια στιγμή θα εμποδίσει τον αλγόριθμο να καλέσει τον εαυτό του. Ο αναδρομικός αλγόριθμος ο οποίος δεν έχει κάποιο κριτήριο τερματισμού παραβιάζει τον ορισμό του αλγορίθμου.
- Η πρώτη εντολή η οποία προκαλεί την αναδρομή δεν ολοκληρώνεται μέχρι την ολοκλήρωση του αναδρομικού τμήματος του αλγόριθμου το οποίο προκάλεσε. Αυτό αληθεύει και τις αναδρομικές κλήσεις οι οποίες προκαλούνται από προηγούμενες αναδρομικές κλήσεις.
- Η αναδρομή υλοποιείται με κάποια εντολή η οποία προκαλεί την εκτέλεση ενός τμήματος του αλγόριθμου ο οποίος συμπεριλαμβάνει και αυτή την εντολή. Κάθε αναδρομική κλήση μιας μεθόδου δημιουργεί νέες τοπικές μεταβλητές και παραμέτρους.
- Η αναδρομή είναι ο καλύτερος τρόπος να λυθούν κάποια προβλήματα, αλλά για άλλα προβλήματα η αναδρομή δημιουργεί πιο πολύπλοκες λύσεις από την επανάληψη (for, while, do). Ο προγραμματιστής είναι αυτός που θα επιλέξει τον πιο κατάλληλο τρόπο.

2. Τυπικές αναδρομικές μέθοδοι

Να υλοποιήσετε τα ακόλουθα προβλήματα με την μέθοδο της αναδρομής.

α) Παραγοντικό.

Το κλασσικό παράδειγμα επίδειξης αναδρομής είναι ο υπολογισμός του παραγοντικού ενός αριθμού. Σημειώνουμε ότι ο υπολογισμός του παραγοντικού ενός αριθμού N (N!) ορίζεται μαθηματικά ως:

$$N! = \begin{cases} N*(N-1)! & \varepsilon\acute{\alpha}\nu \ \mathrm{N} > 0 \\ 1 & \varepsilon\acute{\alpha}\nu \ \mathrm{N} \le 0 \end{cases}$$

Για οποιοδήποτε ακέραιο N, το N παραγοντικό (N!) ορίζεται ως το γινόμενο του N επί το παραγοντικό του N-1. Η αναδρομή σταματάει όταν το N γίνει μικρότερο από ή ίσο με το μηδέν.

β) Ακολουθία Fibonacci

Ένα ακόμη κλασσικό παράδειγμα επίδειξης αναδρομής είναι ο υπολογισμός των αριθμών της ακολουθίας Fibonacci. Κάθε αριθμός ισούται με το άθροισμα των δύο προηγούμενων. Οι δύο πρώτοι είναι ίσοι με τη μονάδα. Έτσι η ακολουθία είναι:

1, 1, 2, 3, 5, 8, 13, 21, …

3. Υπολογισμός Παλίνδρομου αριθμού

Να γραφεί ένα πρόγραμμα που να διαβάζει έναν θετικό ακέραιο (μονοψήφιο έως πενταψήφιο) και να εκτυπώνει μήνυμα για το εάν ο αριθμός αυτός είναι παλίνδρομος ή όχι.

Παλίνδρομος είναι ο αριθμός που διαβάζεται το ίδιο από αριστερά προς δεξιά και το αντίστροφο, δηλαδή ο 121 είναι παλίνδρομος αριθμός ενώ ο 123 δεν είναι παλίνδρομος. Το πρόγραμμα θα πρέπει να εμφανίζει μηνύματα λάθους στις περιπτώσεις που δεν δίνεται από τον χρήστη θετικός έως πενταψήφιος ακέραιος και σε περίπτωση σφάλματος ο χρήστης θα πρέπει να δώσει εκ νέου κάποιον αριθμό.
Το πρόγραμμα σας θα πρέπει να δέχεται 10 εισαγωγές αριθμών από κάθε χρήστη.

Παραλλαγή : Να γραφεί ένα πρόγραμμα που να διαβάζει μία συμβολοσειρά και να εκτυπώνει μήνυμα εάν είναι παλίνδρομος ή όχι.

4. Πρώτοι αριθμοί

Να γραφεί ένα πρόγραμμα που να υπολογίζει εάν ένας αριθμός είναι πρώτος ή όχι.
Πρώτος αριθμός θεωρείται ένα ακέραιος που δεν μπορεί να διαιρεθεί με κάποιο άλλο ακέραιο εκτός από τον εαυτό του και την μονάδα. Για παράδειγμα το 7 είναι πρώτος αριθμός γιατί οι διαιρέτες του είναι το 7 και το 1. Ο ακέραιος 8 δεν είναι πρώτος γιατί οι διαιρέτες του είναι το 1, 2, 4 και 8.

Ο υπολογισμός να γίνει με την μέθοδο της αναδρομής.

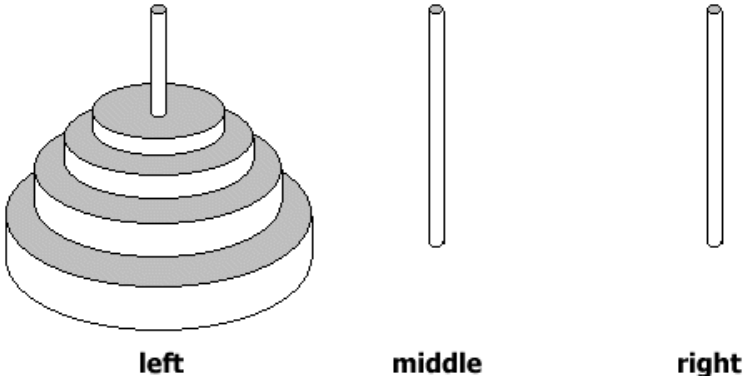
Γράψτε ένα πρόγραμμα που χρησιμοποιεί την παραπάνω μέθοδο και εμφανίζει όλους τους πρώτους αριθμούς από το 1 μέχρι το 10.000.

5. Πύργοι του Ανόϊ

Να γραφεί ένα πρόγραμμα που να επιλύει το ακόλουθο πρόβλημα με την μέθοδο της αναδρομής :

Εχουμε μία κατασκευή με τρείς στύλους. Στον πρώτο είναι περασμένοι **4** δίσκοι. Πρέπει να μεταφέρουμε τους δίσκους από τον πρώτο στύλο στον τρίτο χρησιμοποιώντας τον δεύτερο στύλο σαν βοηθητικό με τους ακόλουθους περιορισμούς:

1. Μόνο ένας δίσκος μπορεί να μετακινηθεί κάθε φορά.
2. Δεν είναι δυνατόν να τοποθετηθεί μεγαλύτερος στύλος πάνω από ένα μικρότερο.



Το πρόγραμμα σας θα εμφανίζει στην οθόνη ακριβείς οδηγίες για μετακίνηση των δίσκων. Ετσι για μετακίνηση τρίων δίσκων από τον 1ο στύλο στον 3ο στύλο το πρόγραμμα πρέπει να γράψει:

1 -> 3
1 -> 2
3 -> 2
1 -> 3
2 -> 1
2 -> 3
1 -> 3

Υποδείξεις:

Μπορείτε να χρησιμοποιήσετε μία βοηθητική αναδρομική μέθοδο με 4 παραμέτρους:

1. Ο αριθμός των δίσκων που θα μετακινηθούν
2. Ο στύλος στον οποίο βρίσκονται οι δίσκοι
3. Ο στύλος στον οποίο θα καταλήξουν οι δίσκοι
4. Ο στύλος που θα χρησιμοποιηθεί σαν βοηθητικός

Παραλλαγή : Τροποποιήστε το πρόγραμμα ώστε να επιλύει το πρόβλημα για 64 δίσκους.

Βοήθεια:

1. Χρησιμοποιείστε την παρακάτω κλάση (ή εμπλουτίστε την υπάρχουσα) για την εισαγωγή Strings από το πληκτρολόγιο, από την οποία θα χρησιμοποιηθεί η μέθοδος getString() για την εισαγωγή του προς αναζήτηση επωνύμου.

```
import java.io.*;

class UserInput {                    //Class gia eisagogi dedomenwn apo to pliktrologio

    public static String getString() {      //Methodos gia eisagogi String
        String line;
        InputStreamReader input=new InputStreamReader(System.in);
        BufferedReader in=new BufferedReader(input);
        try {
            line=in.readLine();
            return line;
        }
        catch(Exception e) {
            return "Exception";
        }
    }

    public static int getInteger() {                //Methodos gia eisagogi Integer
        String line;
        InputStreamReader input=new InputStreamReader(System.in);
        BufferedReader in=new BufferedReader(input);
        try {
            line=in.readLine();
            int i=Integer.parseInt(line);
            return i;
        }
         catch(Exception e) {
            return -1;
        }
    }

}
```