Αντικειμενοστρεφής Προγραμματισμός

Παναγιώτης Αδαμίδης
adamidis@ihu.gr

Αναδρομή (Recursion)

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

Μια φορά και έναν καιρό ήταν ...

- ένα παιδάκι που ...
  - δεν μπορούσε να κοιμηθεί, έτσι η μαμά του, του είπε ένα παραμύθι για ένα αρκουδάκι που ...
    - δεν μπορούσε να κοιμηθεί, έτσι η μαμά του, του είπε ένα παραμύθι για ένα αρνάκι που ...
      - δεν μπορούσε να κοιμηθεί, έτσι η μαμά του, του είπε ένα παραμύθι για ένα σκυλάκι που ...
      - ... κοιμήθηκε
    - ... κοιμήθηκε (αρνάκι)
  - ... κοιμήθηκε (αρκουδάκι)
- ... κοιμήθηκε (παιδάκι)

Αναδρομή (Recursion)

- Κάποια έννοια ή λειτουργία ορίζεται αναδρομικά όταν περιγράφεται χρησιμοποιώντας τον εαυτό της.
- Παράδειγμα:
  Ο ***απόγονος*** ενός ατόμου είναι:
  - ένα παιδί του ή
  - ο ***απόγονος*** ενός παιδιού του.
- Ο ορισμός είναι αναδρομικός γιατί περιγράφεται χρησιμοποιώντας τον εαυτό του.
- Περιλαμβάνει:
  - Συνθήκη τερματισμού (base case)
  - Βήμα προσαρμογής (άτομο → παιδί)
  - Αναδρομική κλήση (μείωση πολυπλοκότητας με κάθε κλήση)

Σημαντικά σημεία 1/2

- Κάθε αναδρομικός αλγόριθμος πρέπει να έχει ένα σημείο τερματισμού του. 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 στις αναδρομικές κλήσεις οι οποίες προκαλούνται από προηγούμενες αναδρομικές κλήσεις.

Σημαντικά σημεία 2/2

- 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.

Παραγοντικό

4! = 1 * 2 * 3 * 4 = 24
x! = 1 * 2 * 3 * … x
fact(x) = x!

- **Συνθήκη τερματισμού:**
  Γνωρίζουμε ότι το παραγοντικό του 0 είναι 1.
- **Αναδρομική κλήση:**
  N! = N * (N-1)!

```
public static double fact (double value)
```

- «Κωδικοποίηση» …στον πίνακα

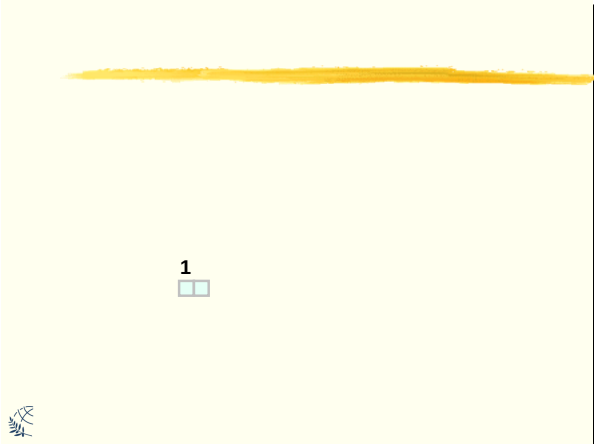
Άλλο παράδειγμα

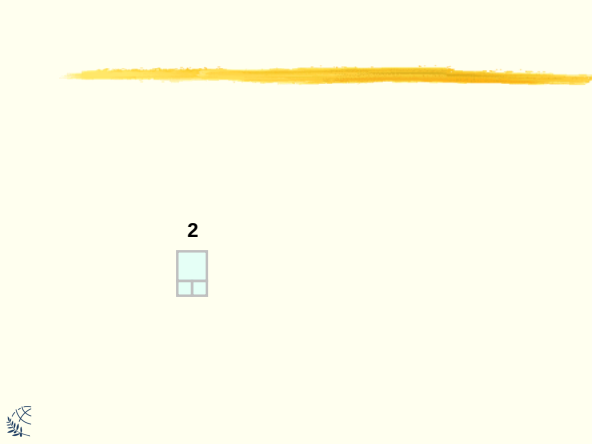
Από τη Γεωμετρία:

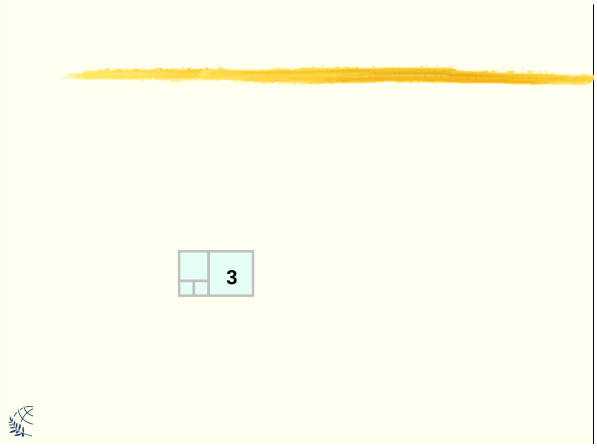
- **Σχεδίαση τετραγώνου με μέγεθος πλευράς 1.**
- **Περιστροφή 90 μοίρες, πρόσθεση τετραγώνου με πλευρά 1.**
- **Περιστροφή 90 μοίρες ξανά, πρόσθεση τετραγώνου με πλευρά 2.**
- **Ξανά, περιστροφή, πρόσθεση τετραγώνου με πλευρά 3, κ.ο.κ..**

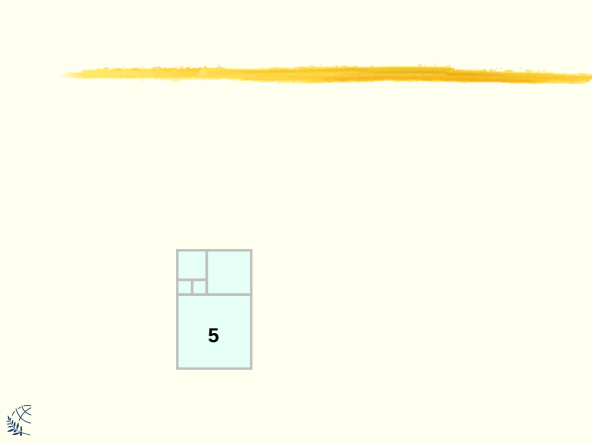
Συνεχίζουμε έτσι για την ακολουθία:

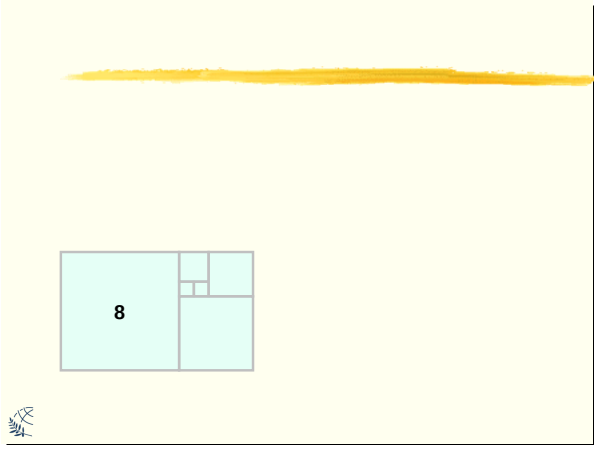
1, 1, 2, 3, 5, 8, 13, 21, . . .

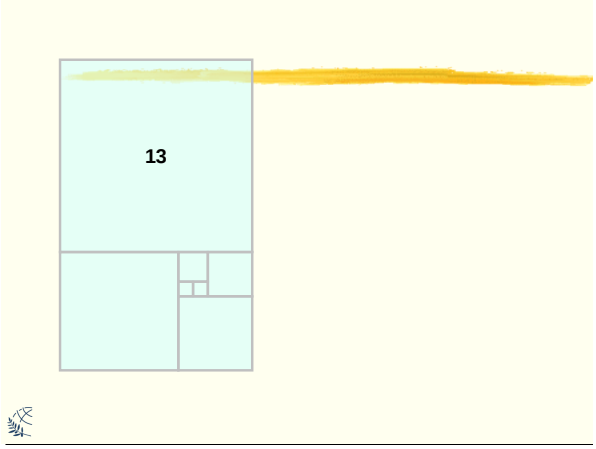


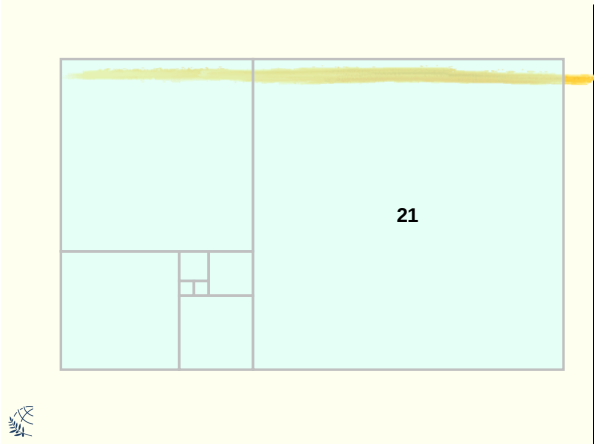


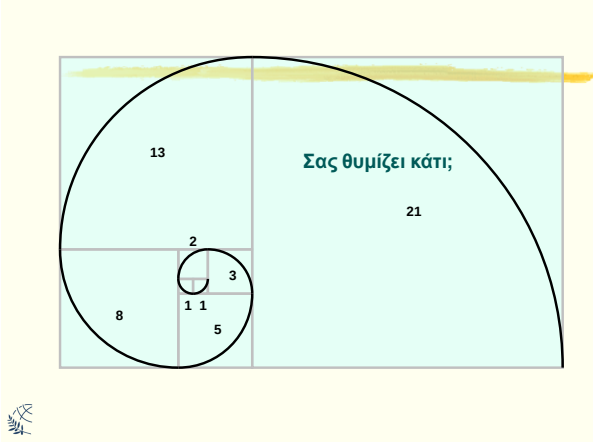


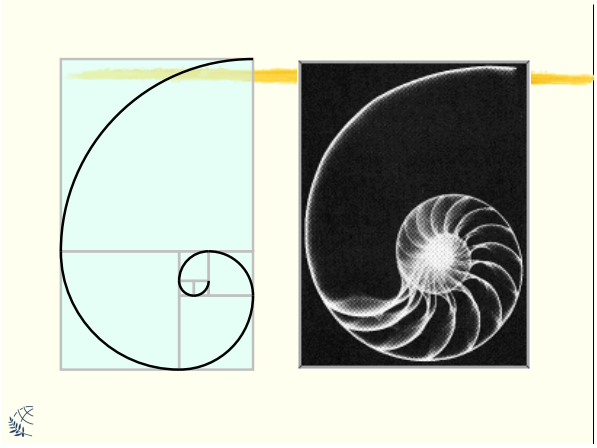



8


13


21


13
Σας θυμίζει κάτι;
21
2
3
1 1
8
5




21
34
Η απάντηση είναι έξω στη φύση

Η ακολουθία αριθμών

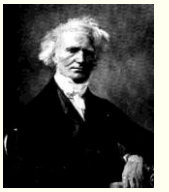
Η ακολουθία αριθμών:
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
αποτελεί την σειρά Fibonacci.

Οι αριθμοί Fibonacci αναφέρονται σε νουβέλες, films, τηλεοπτικές σειρές και τραγούδια. Επίσης στην δημιουργία μουσικής και τέχνης.

Η σειρά επαναλαμβάνεται επίσης σε: κοχύλια, ηλίανθοι, κουκουνάρια, μέλισσες, χρηματιστήριο, κλπ.

Επινόηση τύπου

Δεδομένης της σειράς Fibonacci, μπορούμε να βρούμε έναν τύπο που θα μας δίνει οιοδήποτε όρο, *n* της σειράς;



Jacques Binet (1786-1856)

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

Το επίπεδο των δικών μας γνώσεων μαθηματικών, της εμπειρίας μας, ή ακόμη και της ευφυΐας μας, πολλές φορές, δεν αρκεί. Τι κάνουμε;

Γράφουμε κώδικα.

Κώδικας

- «Κωδικοποίηση» ...στον πίνακα

```
public static int fibo (int num) {
  if (num == 1 || num == 2)
     return 1;
  else
     return fibo(num-1) + fibo(num-2);
}// fibo
```

Ύψωση σε δύναμη

$4^3 = 4 * 4 * 4 = 64$
$x^n = x * x * x * \ldots x$
power(x, n) = x^n

- Συνθήκη τερματισμού: $x^0 = 1$
- Αναδρομική κλήση:

$$x^n = x * x^{n-1}$$

```
public static double power
                (double base, double exp)
```

- «Κωδικοποίηση» ...στον πίνακα

Ανασκόπηση Αναδρομής

1. Υπάρχει στη φύση. Είναι αυτονόητη στα περισσότερα προβλήματα.
2. Είναι ***ευκολότερη*** από την απομνημόνευση κάποιου τύπου.
3. Δεν μπορούν όλα τα προβλήματα να εκφραστούν με κάποιο τύπο, αλλά μπορούν να εκφραστούν σαν μία σειρά μικρών, επαναλαμβανόμενων βημάτων.
4. Κάθε βήμα σε μια αναδρομική διαδικασία πρέπει να είναι μικρό και δυνατό να υπολογιστεί.
5. **Απαραίτητη η συνθήκη τερματισμού.**

Γενικότερα

1. Τα τυπικά παραδείγματα αναδρομής είναι το παραγοντικό (factorial) και οι αριθμοί Fibonacci.
2. Πολλοί τύποι ή ορισμοί είναι ήδη αναδρομικοί – εκμεταλλευτείτε το!
3. Η αναδρομή είναι ιδανική για αλγόριθμους οι οποίοι περιγράφουν αναδρομικές δομές δεδομένων όπως δυαδικά δένδρα. Πολύ πιο εύκολη από την επανάληψη (iteration)!
4. Η αναδρομή είναι ιδανική επίσης για αλγόριθμους «διαίρει και βασίλευε» (divide & conquer).

Παράδειγμα - 1

```
class rec1 {
  public static void main(String[] args){
     A obj = new A();
     obj.meth(5);
  }
}
class  A {
  public void meth(int n) {
     if (n > 0) {
          System.out.println(n);
          meth(n-1);
     }
  }
}
```

Παράδειγμα – 1 (Εκτέλεση)

- obj.meth(5)
  - System.out.println(5)
  - meth(4)
    - System.out.println(4)
    - meth(3)
      - System.out.println(3)
      - meth(2)
        - System.out.println(2)
        - meth(1)
          - System.out.println(1)
          - meth(0)
            - … ΤΕΛΟΣ
          - … ΤΕΛΟΣ meth(1)
        - … ΤΕΛΟΣ meth(2)
    - … ΤΕΛΟΣ meth(3)
  - … ΤΕΛΟΣ meth(4)
- … ΤΕΛΟΣ meth(5)

Έξοδος
5
4
3
2
1

Παράδειγμα - 2

```
class rec1 {
  public static void main(String[] args){
     A obj = new A();
     obj.meth(5);
  }
}
class  A {
  public void meth(int n) {
     if (n > 0) {
          meth(n-1);
          System.out.println(n);
     }
  }
}
```

Παράδειγμα – 2 (Εκτέλεση)

- obj.meth(5) // 5>0
  - meth(4) // 4>0
    - meth(3) // 3>0
      - meth(2) // 2>0
        - meth(1) // 1>0
          - meth(0) // 0=0
            - … ΤΕΛΟΣ
          - System.out.println(1)
        - … ΤΕΛΟΣ meth(1)
        - System.out.println(2)
      - … ΤΕΛΟΣ meth(2)
      - System.out.println(3)
    - … ΤΕΛΟΣ meth(3)
    - System.out.println(4)
  - … ΤΕΛΟΣ meth(4)
  - System.out.println(5)
- … ΤΕΛΟΣ meth(5)

Έξοδος
1
2
3
4
5

Παράδειγμα - 3

```
class rec2 {
  public static void main(String[] args) {
     int x = 1;
     A obj = new A();
     obj.meth(x);
  }
}
class  A {
  public void meth(int x) {
     if (x < 3) {
          meth(++x);
          System.out.println(x);
          meth(x++);
          System.out.println(x);
     }
  }
}
```

Παράδειγμα - 4 (Αμοιβαία αναδρομή)

```
class rec3 {
  public static void main(String[] args) {
     A obj = new A();
     obj.meth1(2);
  }
}
class  A {
  public void meth1(int x) {
     if (x < 5) {
          meth1(++x);
          meth2(++x);
     }
  }
  public void meth2(int x) {
     meth1(x);
     System.out.println(x);
  }
}
```

countChar

➢Δίνεται μια συμβολοσειρά s. Υπολογίστε αναδρομικά το πλήθος των χαρακτήρων ch στη συμβολοσειρά.

```
public static int countChar (String s, char ch)
```

countChar("xxhixx", 'x') → 4
countChar("xhixhix", 'x') → 3
countX("hi", 'x') → 0

➢ «Κωδικοποίηση» ...στον πίνακα

sumDigits

➢Δίνεται ένας μη αρνητικός ακέραιος αριθμός. Η μέθοδος υπολογίζει το άθροισμα των ψηφίων του.

```
public static int sumDigits (int n)
```

sumDigits(126) → 9
sumDigits(49) → 13
sumDigits(88512) → 24

```
public static int sumDigits(int n) {
  if (n<10) return n;
  return n%10+sumDigits(n/10);
}
```

reverseNumberDisplay

➢Δίνεται ένας μη αρνητικός ακέραιος αριθμός. Η μέθοδος φανίζει τα ψηφία του αριθμού με αντίστροφη σειρά.

```
public static void reverseNumberDisplay(int n)
```

reverseDisplay(12345) → 54321

```
public static void revNumberDisplay (int n){
      if (n<10)
           System.out.println(n);
      else  {
           System.out.println(n%10);
           revNumberDisplay(n/10);
           }
}
```

Ερωτήσεις;