



Διάλεξη 15: Αναδρομή (Recursion)

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

- Η έννοια της αναδρομής
- Μη-αναδρομικός / Αναδρομικός Ορισμός Συναρτήσεων
- Παραδείγματα Ανάδρομης
- Αφαίρεση της Αναδρομής

Μη αναδρομικές συναρτήσεις

- Προτού δούμε τι είναι αναδρομή θεωρήστε το πρόβλημα εύρεσης του παραγοντικού κάποιου αριθμού (factorial).

$$0! = 1, \quad 1! = 1 \quad 2! = 1 \times 2 = 2 \quad 3! = 1 \times 2 \times 3 = 6$$

$$4! = 1 \times 2 \times 3 \times 4 = 24 \quad 5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

***Ζητούμενο:** Να υλοποιήσουμε την `factorial(int n)` η οποία μας επιστρέφει το παραγοντικό κάποιου θετικού ακεραίου n .*

Λύση

```
int factorial(int n) {  
    int i, result=1;  
    for (i=1; i<=n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

Αναδρομή

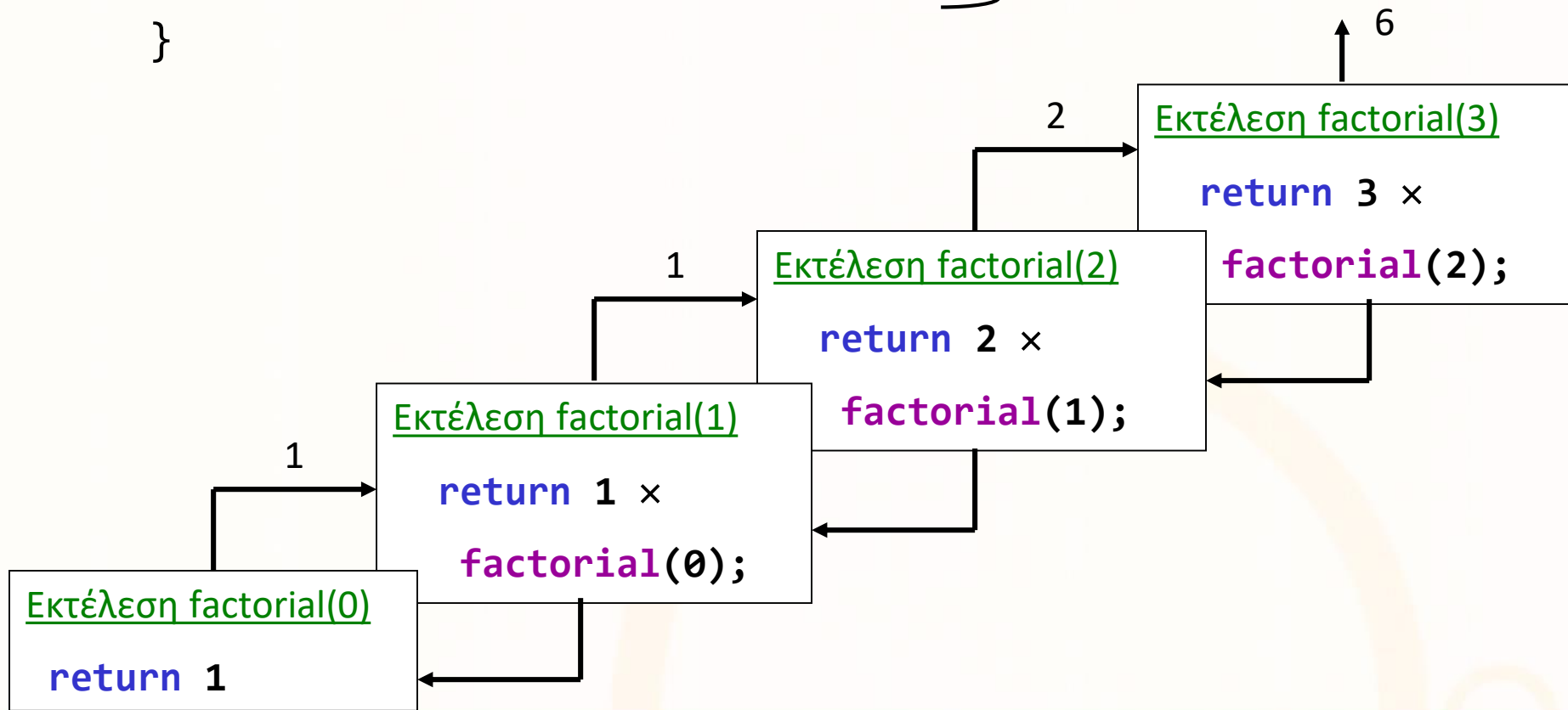
- **Βασική έννοια στα Μαθηματικά και στην Πληροφορική.**
- Στην πληροφορική η αναδρομή χρησιμοποιείται σαν *τεχνική προγραμματισμού* και σαν *μέθοδος σχεδιασμού αλγορίθμων*.
- Στον προγραμματισμό η αναδρομή εμφανίζεται με την **κλήση ενός υποπρογράμματος από τον εαυτό του**. Ένα αναδρομικό υποπρόγραμμα αποτελείται από:
 - ένα **βήμα τερματισμού**, όπου ορίζεται η εκτέλεση του υποπρογράμματος για κάποιες “μικρές” τιμές των παραμέτρων του, και
 - ένα **αναδρομικό βήμα**, κατά το οποίο η εκτέλεση του υποπρογράμματος ορίζεται ως συνδυασμός κλήσεων του υποπρογράμματος σε άλλες “μικρότερες” τιμές των παραμέτρων.

Παράδειγμα 1: Παραγοντικός Αριθμός με Αναδρομή

- Ας ορίσουμε τώρα τον αναδρομικό ορισμό της factorial.
 $0! = 1$, $1! = 1$ $2! = 1 \times 2 = 2$ $3! = 1 \times 2 \times 3 = 6$
 $4! = 1 \times 2 \times 3 \times 4 = 24$ $5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$
- Ας ορίσουμε τώρα τον αναδρομικό ορισμό της factorial.
- **Αναδρομικός Ορισμός συνάρτησης <factorial>**
 - Βήμα Τερματισμού: $0! = 1$
 - Αναδρομικό Βήμα: $n! = n \times (n-1)!$

Παραγοντικός Αριθμός με Αναδρομή

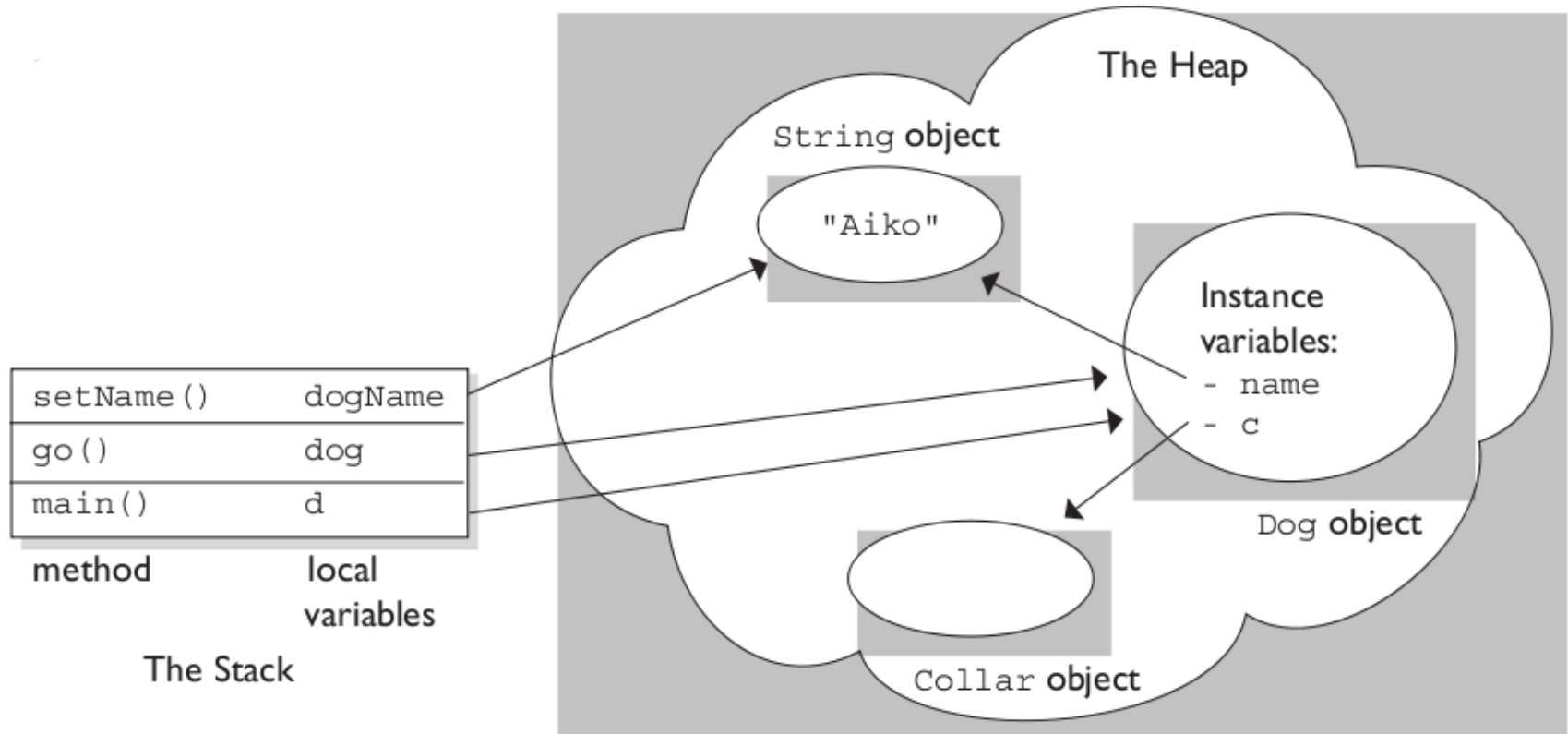
```
int factorial ( int n ) {  
    if (n == 0) } Βήμα Τερματισμού  
        return 1;  
    else } Αναδρομικό Βήμα  
        return n x factorial(n-1);  
}
```



Υλοποίηση αναδρομής

- Σε κάθε κλήση οποιασδήποτε συνάρτησης ένα σύνολο από λέξεις (**stack frame**) φυλάσσεται σε μια **στοίβα (την στοίβα του προγράμματος)**, από όπου μπορεί να ανασυρθεί.
- Όταν μια συνάρτηση διακόψει την εκτέλεσή της με την κλήση μιας άλλης συνάρτησης **οι παράμετροι της συνάρτησης, η διεύθυνση επιστροφής και οι τοπικές μεταβλητές της καλούσας συνάρτησης** φυλάσσονται μέσα στη στοίβα του προγράμματος.
- Έτσι όταν η κληθείσα συνάρτηση τερματίσει το περιβάλλον την καλούσας συνάρτησης **ανασύρεται από τη στοίβα** για να συνεχιστεί κανονικά η εκτέλεσή της.
- Αφού κάθε κλήση μιας διαδικασίας εκτελείται στο δικό της περιβάλλον, είναι επιτρεπτή και **η κλήση συναρτήσεων από τον εαυτό τους (αναδρομή)**.

Stack and Heap (Στοιίβα και Σωρός)



Stack and Heap (Στοιίβα και Σωρός)

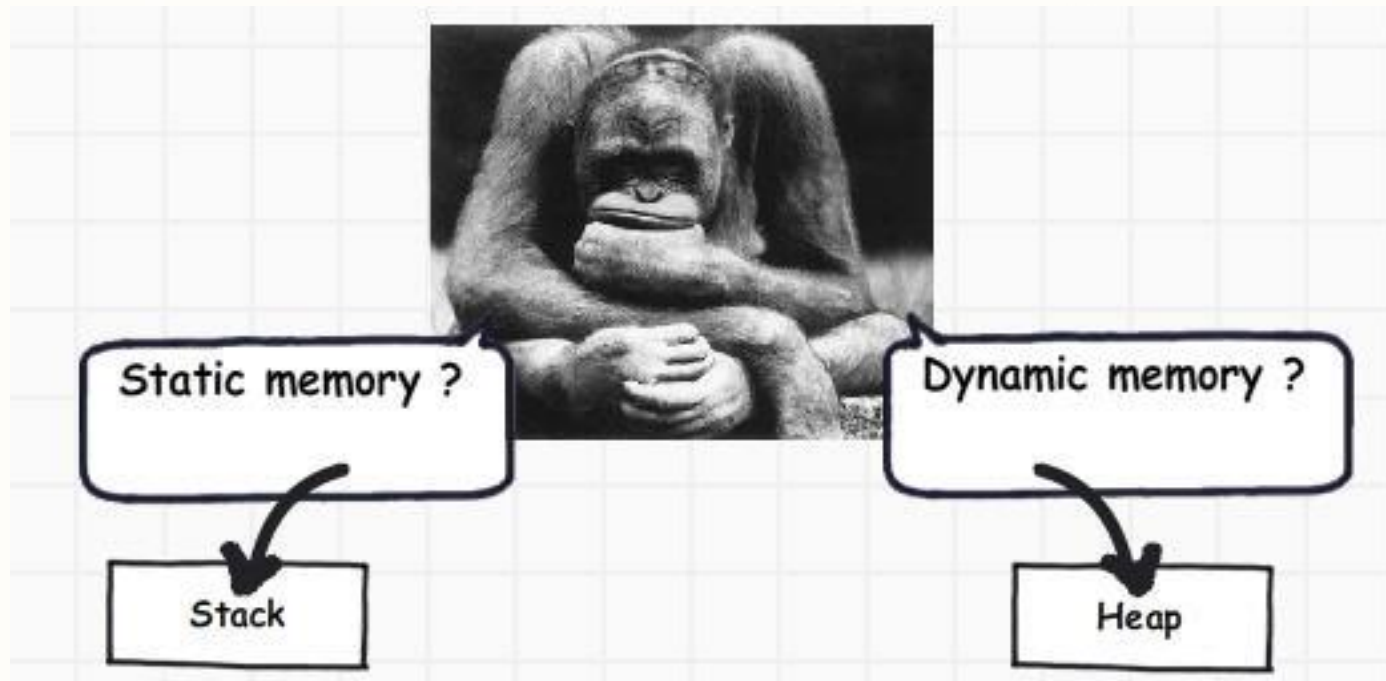


Image from: <https://www.codeproject.com/Articles/76153/Six-important-NET-concepts-Stack-heap-value-types#Stack%20and%20Heap>

Call Stack

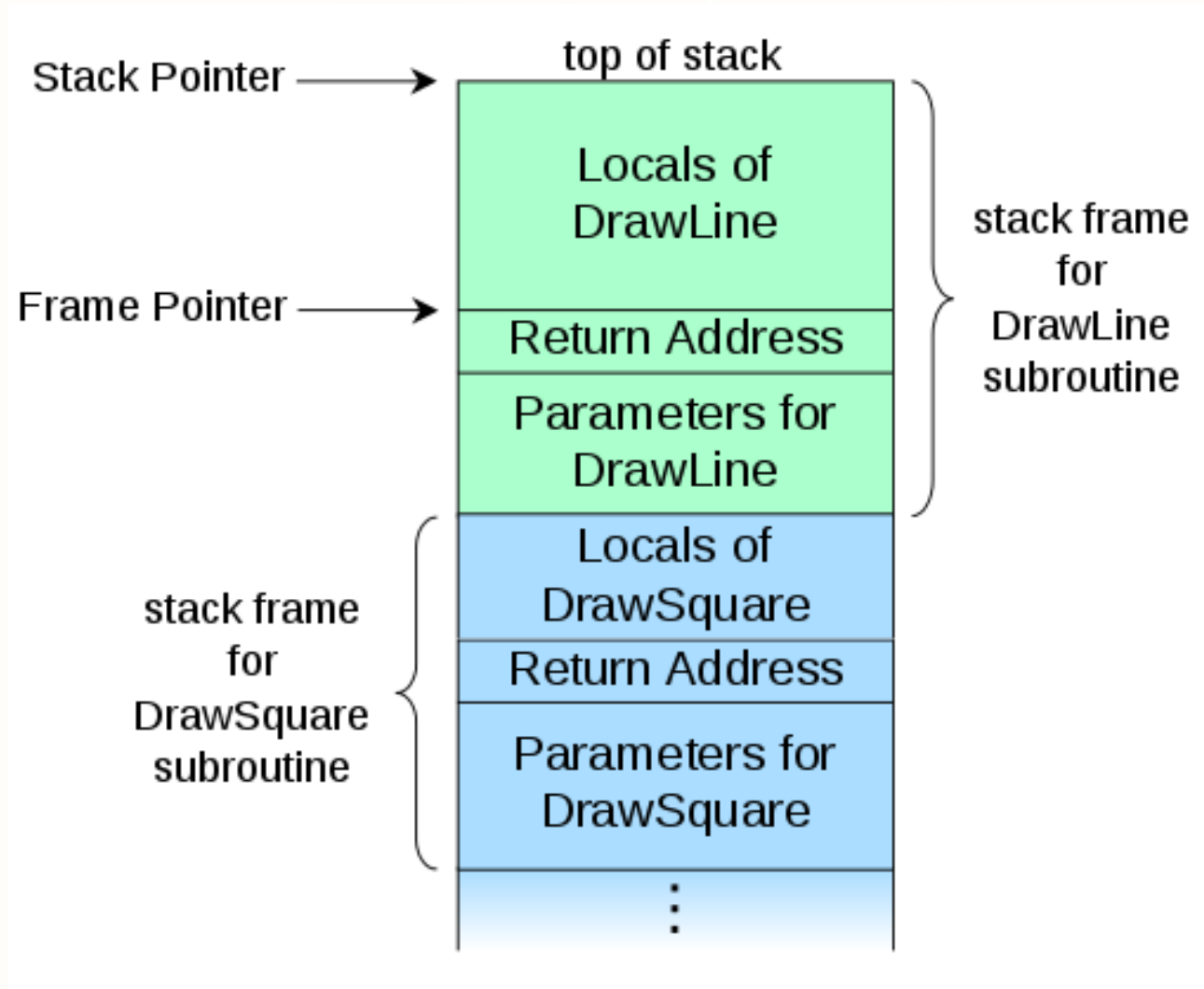


Image from: https://en.wikipedia.org/wiki/Call_stack

Αναδρομή και Διαχείριση Μνήμης

Στοίβα

factorial(2)

main

factorial(2)

return 2 * factorial(1)

factorial(2)

main

factorial(2)

return 2 * factorial(1)

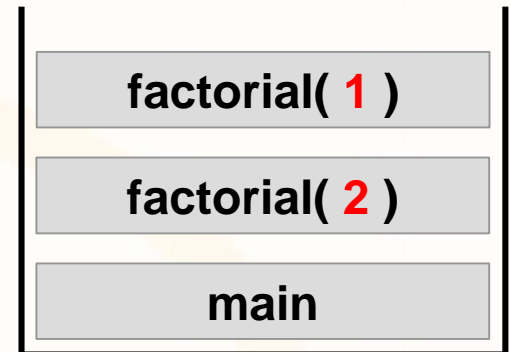
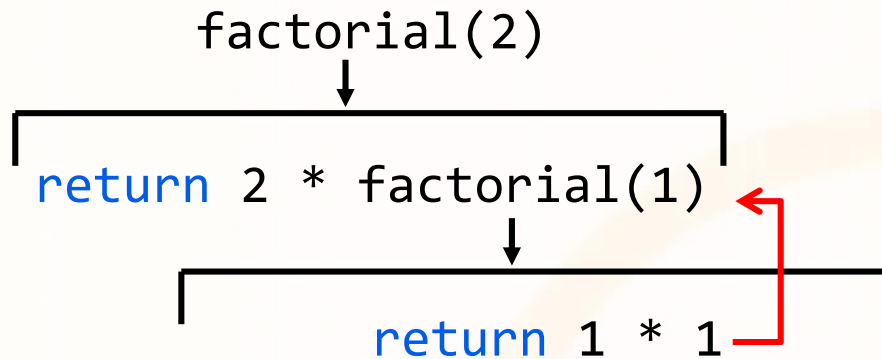
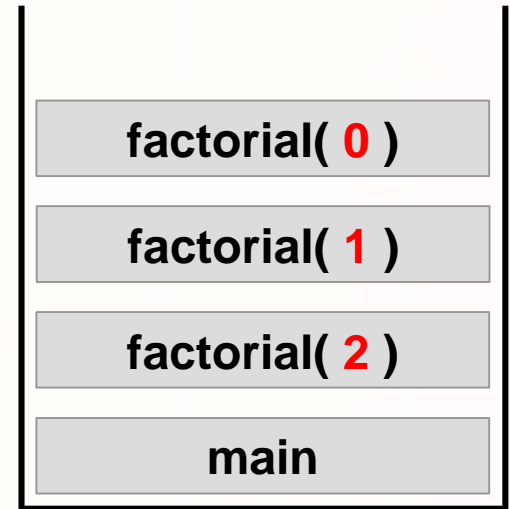
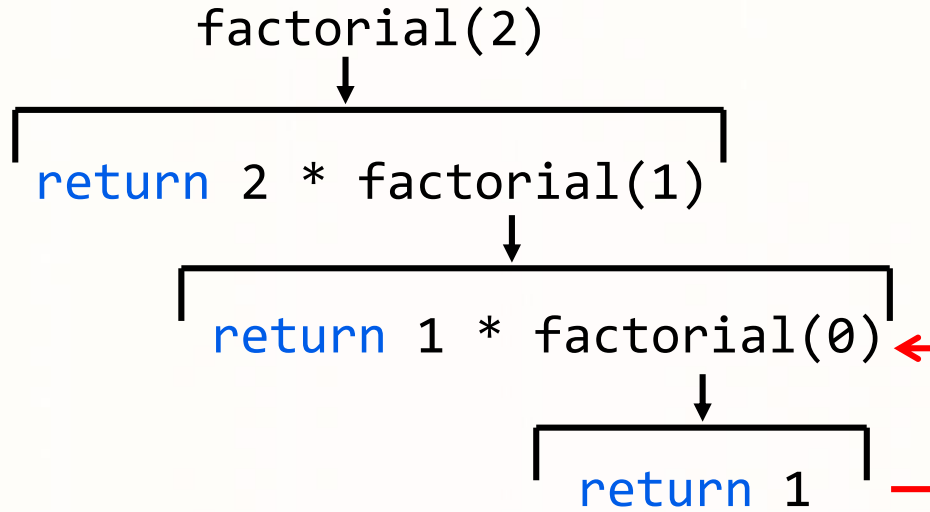
return 1 * factorial(0)

factorial(1)

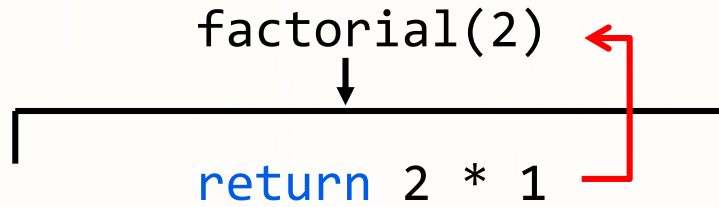
factorial(2)

main

Αναδρομή και Διαχείριση Μνήμης



Αναδρομή και Διαχείριση Μνήμης



Στοίβα

2



Παράδειγμα 2: Δύναμη Αριθμού με Αναδρομή

2) Δύναμη (Power)

$$a^0 = 1$$

$$a^n = a^{n-1} \cdot a \quad (n \geq 1)$$

```
int mpower(int a, int n) {  
    if (n==0) return 1;  
    return a*mpower(a, n-1);  
}
```

Παράδειγμα

$$\begin{aligned} \text{mpower}(2,3) & \Rightarrow 2 * \text{mpower}(2,2) \\ & = 2 * 2 * \text{mpower}(2,1) \\ & = 2 * 2 * 2 * \text{mpower}(2,0) \\ & = 2 * 2 * 2 * 1 = 8 \end{aligned}$$

Παράδειγμα 3: Fibonacci Numbers

3) Αριθμοί Fibonacci (Leonardo of Pisa – 1202μΧ)

Χρησιμοποιήθηκαν για να εκφράσουν την αύξηση κουνελιών!

Σε ιδανικές συνθήκες!

- Στον μήνα 0 έχουμε 0 ζεύγη , και στον μήνα 1 έχουμε 1
- Το ζεύγος γονιμοποιείται μετά τον πρώτο μήνα (δηλ. στον δεύτερο).
- Κάθε μήνα, κάθε ζεύγος παράγει ένα νέο ζεύγος
- Έστω ότι είμαστε στον μήνα **n** και έχουμε ένα πληθυσμό **F(n)** **ζευγών**. Αυτή την στιγμή μόνο κουνέλια που ήταν ζωντανά την στιγμή **n-2** παράγουν ένα νέο ζεύγος.
- Επομένως **F(n-2)** ζευγάρια προστίθεται στον παρόν πληθυσμό των F(n-1).
- Ο ολικός πληθυσμός την στιγμή F(n) είναι επομένως **F(n) = F(n-1) + F(n-2)**

0,1,1,2,3,5,8,13,21,34,55,89,....

Κουνέλια a la Fibonacci

December

Young black couple



January

Black couple now adult



February

Red twins for Black couple



March

Blue twins for Black couple



April

Twins for Black, twins for Red



May

Twins for Black, Red, Blue couples



Image from: http://oldeuropeanculture.blogspot.com/2018/02/fibonacci_24.html

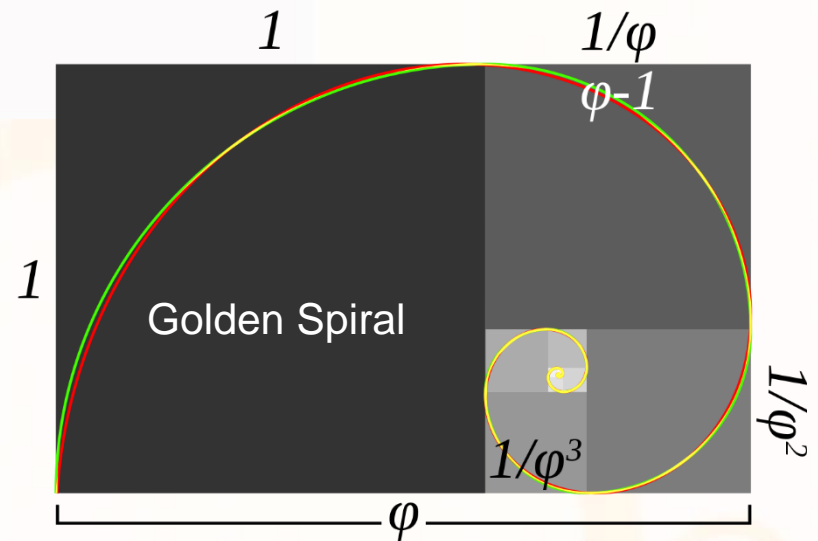
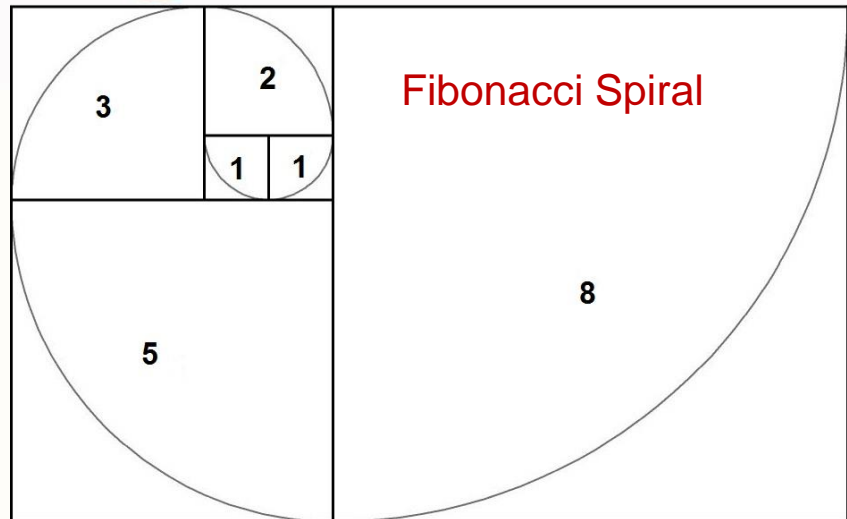
Golden Ratio, Golden Spiral and Fibonacci

$\Phi = 1.6180339$

Current #	Previous #	Division	Ratio
1	1	1 / 1	1
2	1	2 / 1	2
3	2	3 / 2	1.5
5	3	5 / 3	1.6666
8	5	8 / 5	1.6
13	8	13 / 8	1.625
21	13	21 / 13	1.615384
34	21	34 / 21	1.619048

$$\frac{F(n)}{F(n-1)} \rightarrow \varphi$$

http://oldeuropeanculture.blogspot.com/2018/02/fibonacci_24.html



<https://commons.wikimedia.org/wiki/File:FakeRealLogSpiral.png>

Παράδειγμα 3: Fibonacci Numbers (συν.)

- Μαθηματικός Ορισμός

$$F_n = F(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F(n-1) + F(n-2) & n > 1 \end{cases}$$

- Κώδικας 1

```
int fibonacci(int n) {  
    if (n==0)          return 0;  
    else if (n==1)     return 1;  
    else return  
        fibonacci(a-1)  
        + fibonacci(a-2);  
}
```

- Κώδικας 2 😊

```
int fibonacci(int n) {  
    return (n<=1)? n : Fibonacci(n-1) + Fibonacci(n-2);  
}
```

Παράδειγμα 4: Παλίνδρομα

4) Παλίνδρομα (Palindrome)

abac	false
abba	true
aba	true

```
boolean isPalindrome (String s) {  
    if (s.length()<=1)  
        return true;  
  
    else if ( s.charAt(0)!=s.charAt(s.length()-1) )  
        return false;  
  
    else  
        return isPalindrome( s.substring(1, s.length()-1) );  
}
```

Παράδειγμα 4: Παλίνδρομα (συν.)

- Η προηγούμενη μέθοδος παρουσιάζει **πρόβλημα αποδοτικότητας** διότι δημιουργεί ένα καινούριο String κάθε φορά που καλείται αναδρομικά

```
return isPalindrome( s.substring(1, s.length()-1) );
```

- Μερικές φορές είναι καλύτερα να χρησιμοποιηθεί μία βοηθητική συνάρτηση για να αποφύγουμε τέτοια προβλήματα

```
boolean isPalindrome (String s) {  
    return isPalindrome(s, 0, s.length()-1);  
}
```

**Υπερφόρτωση
(Overloading)**

```
boolean isPalindrome (String s, int low, int high) {  
    if (high<=low) return true;  
    else if ( s.charAt(low)!=s.charAt(high) )  
        return false;  
    else  
        return isPalindrome(s, low+1, high-1 );  
}
```

Παράδειγμα 5: Μέγεθος Φακέλου (Directory Size)

```
import java.io.File;
import java.util.Scanner;

public class DirectorySize {
    public static void main(String[] args) {
        // Prompt the user to enter a directory or a file
        System.out.print("Enter a directory or a file: ");
        Scanner input = new Scanner(System.in);
        String directory = input.nextLine();

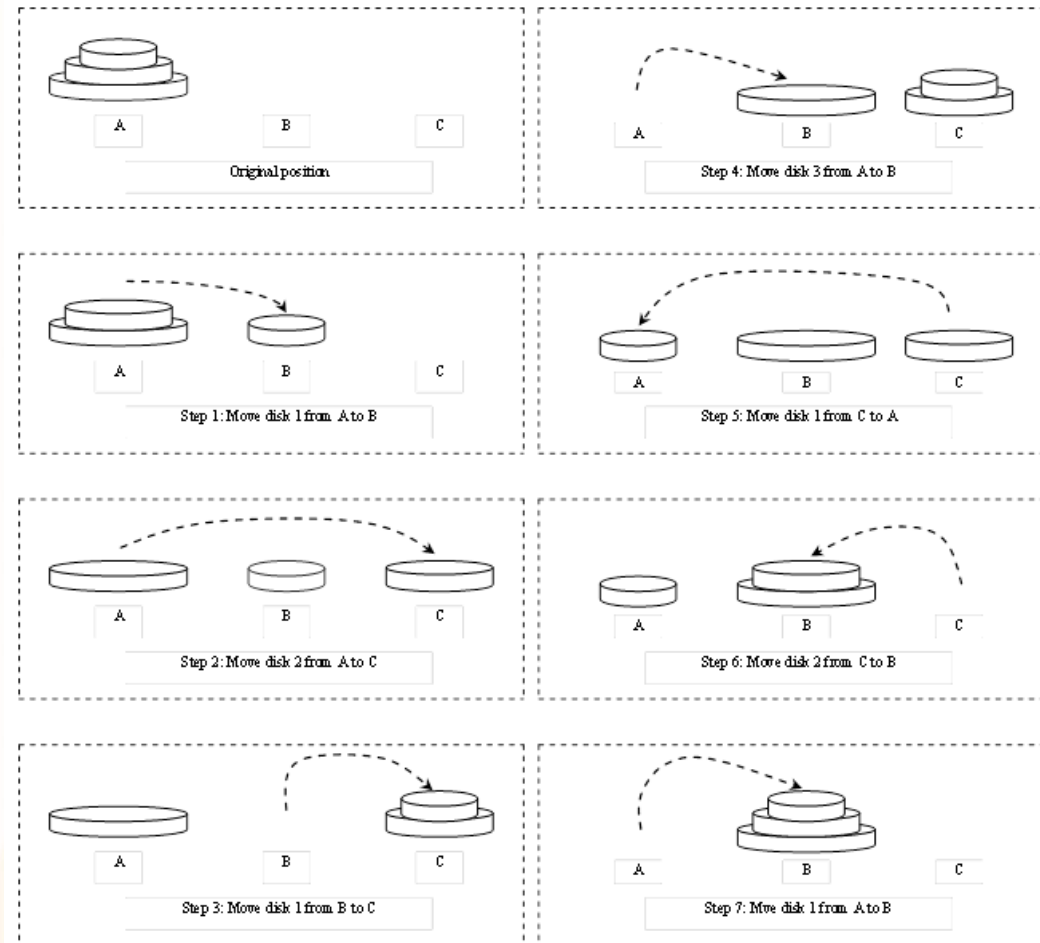
        // Display the size
        System.out.println(getSize(new File(directory)) + " bytes");
    }

    public static long getSize(File file) {
        long size = 0; // Store the total size of all files

        if (file.isDirectory()) {
            File[] files = file.listFiles(); // All files and subdirectories
            for (int i = 0; i < files.length; i++) {
                size += getSize(files[i]); // Recursive call
            }
        }
        else { // Base case
            size += file.length();
        }
        return size;
    }
}
```

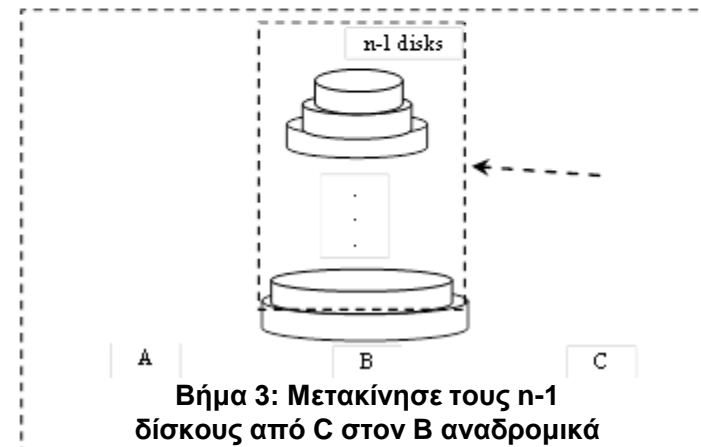
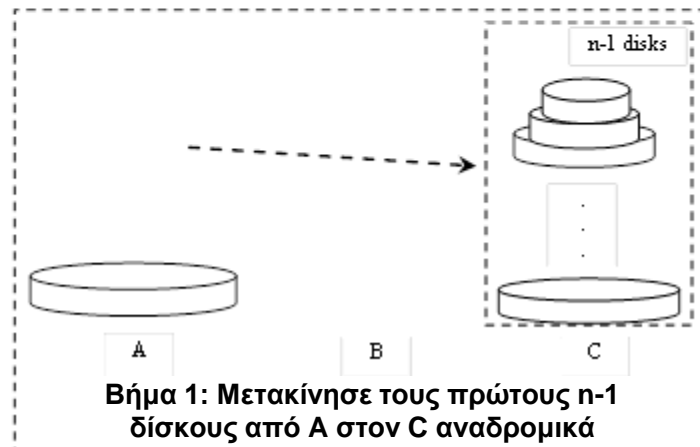
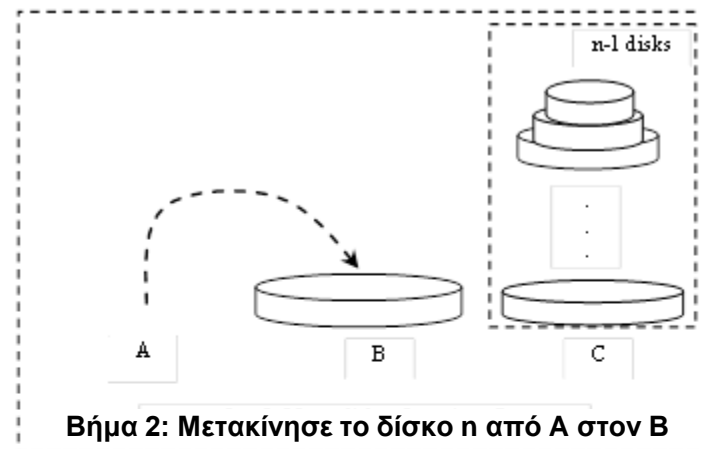
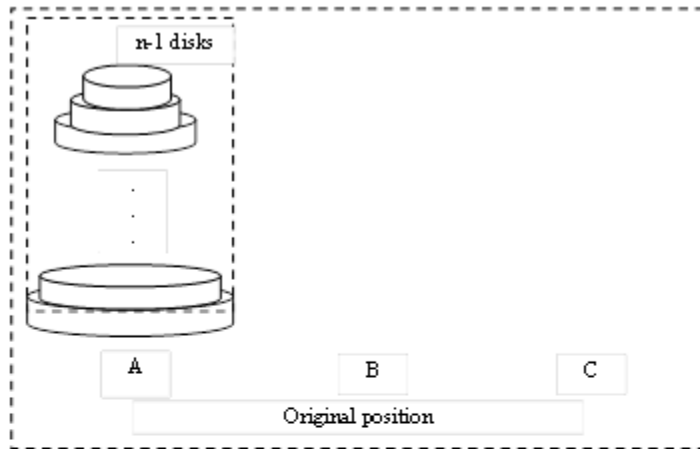
Παράδειγμα 6: Πύργοι του Ηανοί

- Υπάρχουν n δίσκοι με αριθμούς $1, 2, 3, \dots, n$, και τρεις πύργοι με ονόματα A, B, και C.
- Κανένας δίσκος δεν μπορεί να είναι πάνω σε μικρότερο δίσκο ανά πάσα στιγμή.
- Αρχικά, όλοι οι δίσκοι τοποθετούνται στον πύργο A.
- Μόνο ένας δίσκος μπορεί να μετακινηθεί ανά πάσα στιγμή και πρέπει να είναι ο πιο ψηλός δίσκος σε ένα πύργο



Παράδειγμα 6: Πύργοι του Ηanoi (συν.)

- Το πρόβλημα των πύργων του Ηanoi μπορεί να διασπαστεί σε 3 υπο-προβλήματα



Παράδειγμα 6: Πύργοι του Hanoi (συν.)

```
import java.util.Scanner;
public class TowersOfHanoi {

    public static void main(String[] args) {
        // Create a Scanner
        Scanner input = new Scanner(System.in);
        System.out.print("Enter number of disks: ");
        int n = input.nextInt();
        // Find the solution recursively
        System.out.println("The moves are:");
        moveDisks(n, 'A', 'B', 'C');
    }

    /** The method for finding the solution to move n disks
        from fromTower to toTower with auxTower */
    public static void moveDisks(int n, char fromTower,
        char toTower, char auxTower) {
        if (n == 1) // Stopping condition
            System.out.println("Move disk " + n + " from " +
                fromTower + " to " + toTower);
        else {
            moveDisks(n - 1, fromTower, auxTower, toTower);
            System.out.println("Move disk " + n + " from " +
                fromTower + " to " + toTower);
            moveDisks(n - 1, auxTower, toTower, fromTower);
        }
    }
}
```

Αφαίρεση της Αναδρομής

- Η χρήση της αναδρομής επιτρέπει την επίλυση πολύπλοκων προβλημάτων με **άμεσο και σαφή τρόπο**. Συχνά όμως **υστερεί από άποψη αποδοτικότητας**.
- Η **αφαίρεση της αναδρομής από μια συνάρτηση**, δηλαδή, η μετατροπή της σε επαναληπτική συνάρτηση χωρίς αναδρομή, είναι δυνατή (κάτω από κάποιες συνθήκες).
- Συχνά προϋποθέτει τη χρήση κάποιων βοηθητικών δομών (π.χ. στοίβα ή ουρά).

Πρόβλημα 1: Μετατροπή Δεκαδικού σε Δυαδικό

- **Συνάρτηση `int2Binary`:** εκτυπώνει την δυαδική αναπαράσταση ενός φυσικού αριθμού που δίδεται ως παράμετρος στην συνάρτηση.
- Πρότυπο συνάρτησης: **`void int2Binary(int a)`**
- Παραδείγματα:

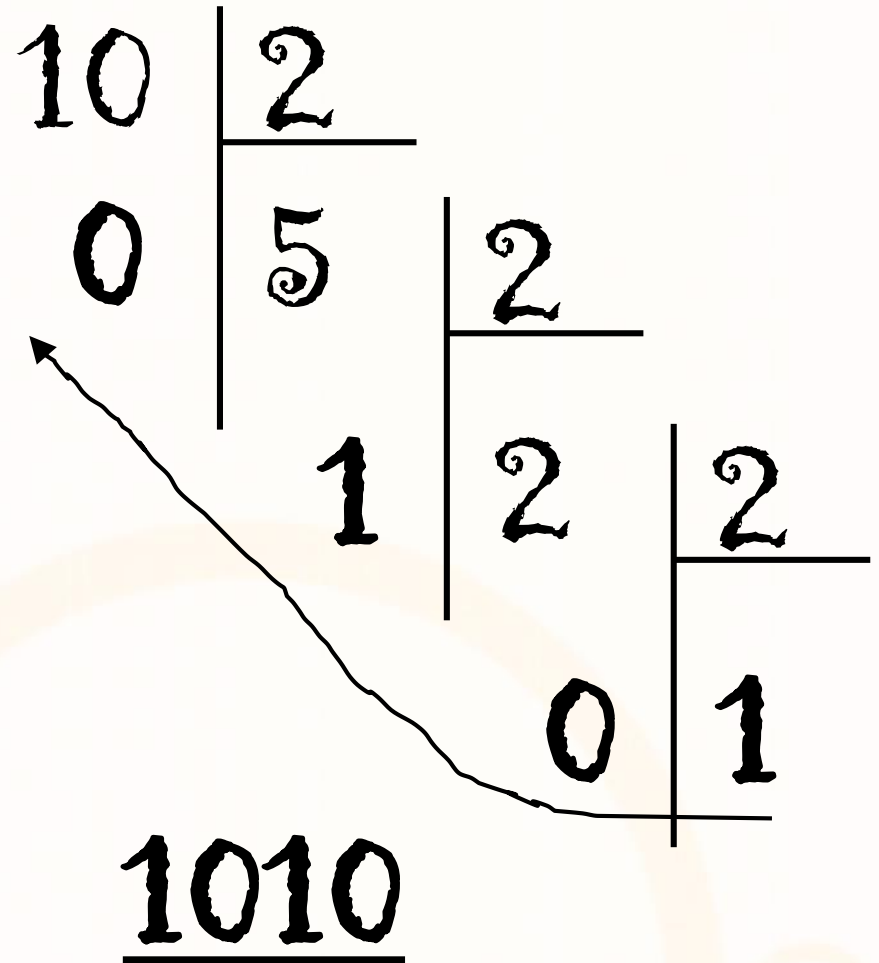
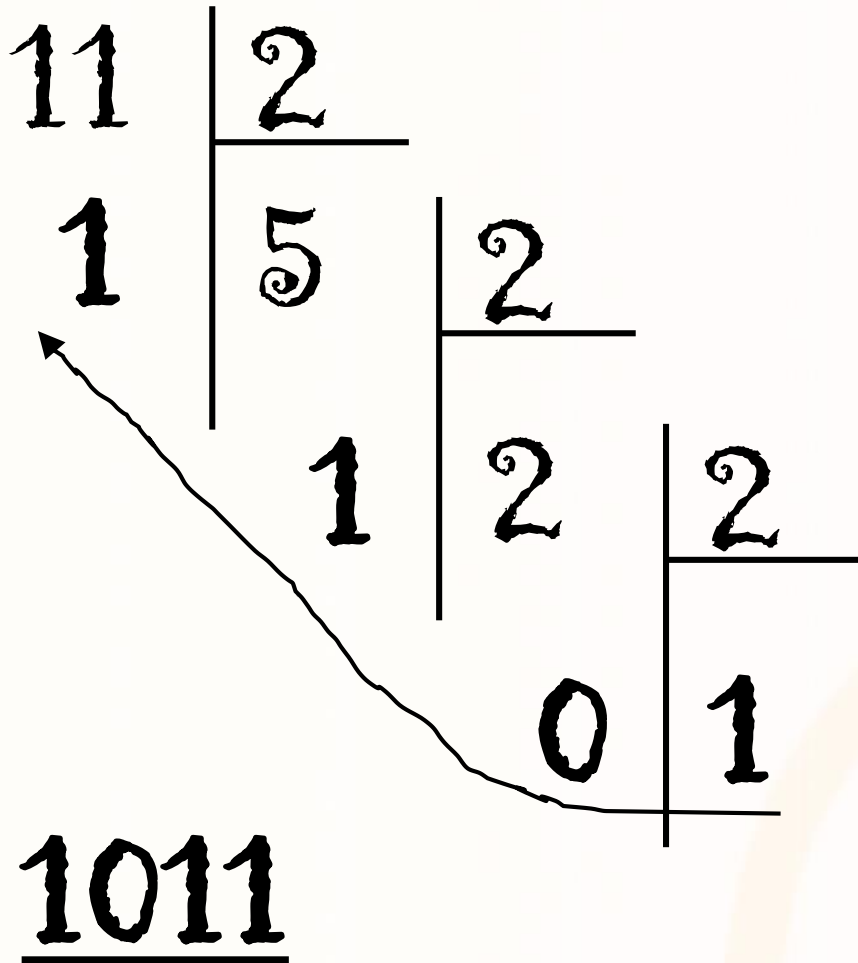
<code>int2Binary(5)</code>	τυπώνει στην οθόνη	101
<code>int2Binary(11)</code>	τυπώνει στην οθόνη	1011
<code>int2Binary(15)</code>	τυπώνει στην οθόνη	1111
<code>int2Binary(124)</code>	τυπώνει στην οθόνη	1111100

Πρόβλημα: Μετατροπή Δεκαδικού σε Δυαδικό(συν.)

- Παράδειγμα Υπολογισμού:

int2Binary(11) → 1011

int2Binary(10) → 1010

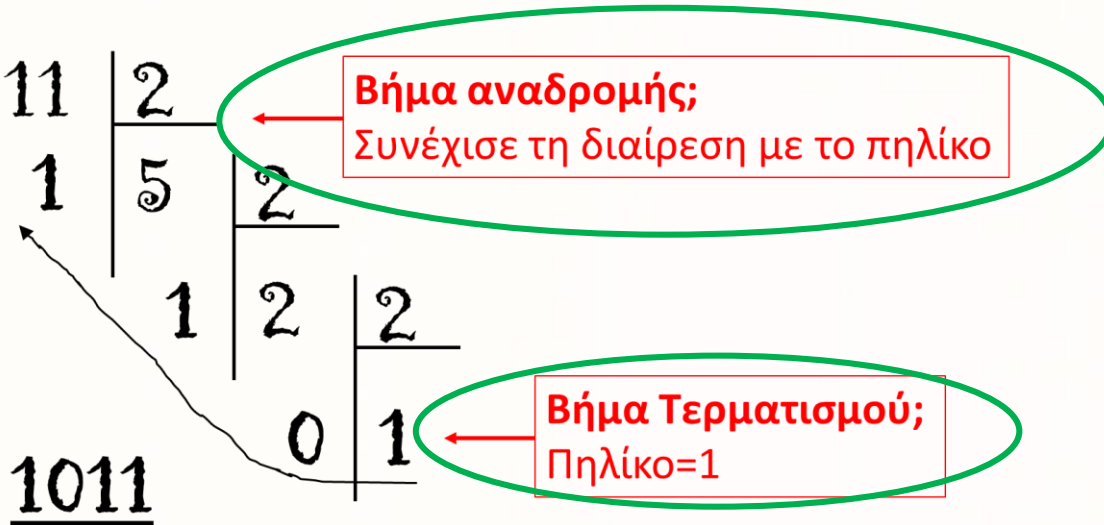


Υλοποίηση συνάρτησης int2Binary: Α. Αναδρομή



Τι κάνουμε στο τέλος;
Τυπώνουμε το πηλίκο=1 και όλα τα υπόλοιπα

Υλοποίηση συνάρτησης int2Binary: Α. Αναδρομή



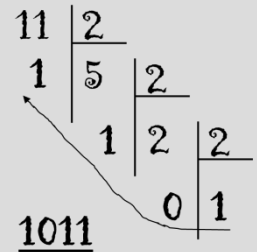
```
void int2Binary_R( int x ){  
    if ( x > 1 ) int2Binary_R(x/2);  
    System.out.print(x%2);  
}
```

Υλοποίηση συνάρτησης int2Binary: Β. Στοίβα

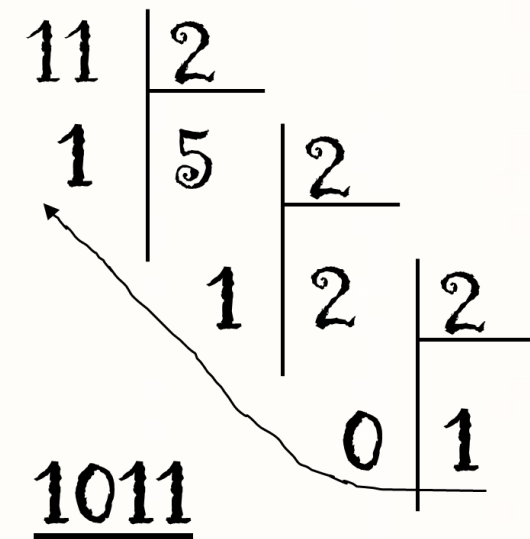
Βήματα

1. Αρχικοποίησε Στοίβα S
2. Όσο ο διαιρετέος είναι μεγαλύτερος του 1
 - A. Πρόσθεσε το υπόλοιπο της διαίρεσης στην S
 - B. Διαίρεσε το διαιρετέο διά 2
3. Βάλε το τελευταίο ψηφίο στην S
4. Όσο δεν είναι άδεια η S
 - A. Εκτύπωσε το στοιχείο στη θέση Top
 - B. Αφαίρεσε το στοιχείο στη θέση Top (Pop)

```
void int2Binary_S(int x) {  
  
    Stack<Integer> stack = new  
        Stack<Integer>();  
    stack.makeEmpty();  
  
    while (x > 1) {  
        stack.push(x % 2);  
        x = x / 2;  
    }  
    stack.push(x);  
  
    while (!stack.isEmpty()) {  
        System.out.print(  
            stack.top());  
        stack.pop();  
    }  
}
```



Υλοποίηση συνάρτησης int2Binary: Γ. Συνάρτηση



```
void int2Binary(int x) {  
  
    int binary[] = new int[33];  
    int i = 0;  
  
    while (x > 1) {  
        binary[i] = x % 2;  
        x = x / 2;  
        i++;  
    }  
    binary[i] = x;  
  
    for (int j = i; j >= 0; j--)  
        System.out.print(binary[j]);  
}
```

Πρόβλημα: Ενδοθεματική Διάσχιση ΔΔΑ

Να γράψετε μη-αναδρομική μέθοδο η οποία να τυπώνει τα στοιχεία ενός ΔΔΑ σε αύξουσα σειρά. Για να το πετύχετε μπορείτε να χρησιμοποιήσετε στοίβες.

- Η βασική ιδέα είναι η εξής:
Εφόσον το δένδρο δεν είναι κενό προχωρούμε προς τα αριστερά φυλάγοντας τους κόμβους από τους οποίους περνούμε σε μία στοίβα.
Όταν δεν μπορούμε να προχωρήσουμε άλλο ανασύρουμε τον κόμβο κορυφής της στοίβας, τυπώνουμε το στοιχείο του, και επαναλαμβάνουμε την ίδια διαδικασία στο δεξιό του παιδί.
- Υποθέτουμε την ύπαρξη υλοποίησης στοίβας και συγκεκριμένα των πράξεων, `MakeEmpty`, `IsEmpty`, `Top`, `Pop` και `Push`.

Ενδοθεματική Διάσχιση ΔΔΑ: Αναδρομική Μέθοδος

```
private void inOrder(BinaryTreeNode<E> node) {  
    if (node != null) {  
        inOrder(node.left);  
        System.out.print(node.key + " ");  
        inOrder(node.right);  
    }  
}
```

Ενδοθεματική Διάσχιση ΔΔΑ: Μη Αναδρομική Μέθ.

```
public void inOrder_S(BinaryTreeNode<E> node) {  
    Stack<BinaryTreeNode<E>> s = new Stack<BinaryTreeNode<E>>();  
    s.makeEmpty();  
    while(node!=null || !s.isEmpty()) {  
        if(node!=null) {  
            s.push(node);  
            node = node.left;  
        }  
        else{  
            node = s.top();  
            s.pop();  
            System.out.print(node.key + " ");  
            node = node.right;  
        }  
    }  
}
```

Αναδρομή/Αφαίρεση Αναδρομής: Ασκήσεις

- Να γράψετε αναδρομική μέθοδο η οποία να τυπώνει τα στοιχεία ενός 2-3 δέντρου σε αύξουσα σειρά.

Αναδρομή/Αφαίρεση Αναδρομής: Ασκήσεις

- Να γράψετε μή-αναδρομική μέθοδο η οποία να τυπώνει τα στοιχεία ενός 2-3 δέντρου σε αύξουσα σειρά.