

Προγραμματισμός I (HY120)

Διάλεξη 16:
Εισαγωγή στην Αναδρομή



Αναδρομικές Συναρτήσεις



2

- Μια συνάρτηση ονομάζεται **αναδρομική** όταν **καλεί τον εαυτό της** – άμεσα ή έμμεσα (μέσα από άλλες συναρτήσεις).
- Μια αναδρομική συνάρτηση πρέπει να τερματίζει.
 - Η ατέρμονη αναδρομή είναι προγραμματιστικό λάθος (αντίστοιχο με αυτό της ατέρμονης επανάληψης) και οδηγεί σε τερματισμό του προγράμματος, λόγω υπερχείλισης της στοίβας (stack overflow).
- Η αναδρομή μπορεί να θεωρηθεί σαν μια ειδική τεχνική προγραμματισμού
 - Διάφορα (πολύπλοκα) προβλήματα μπορεί να λυθούν με φυσικό τρόπο χρησιμοποιώντας αναδρομή.

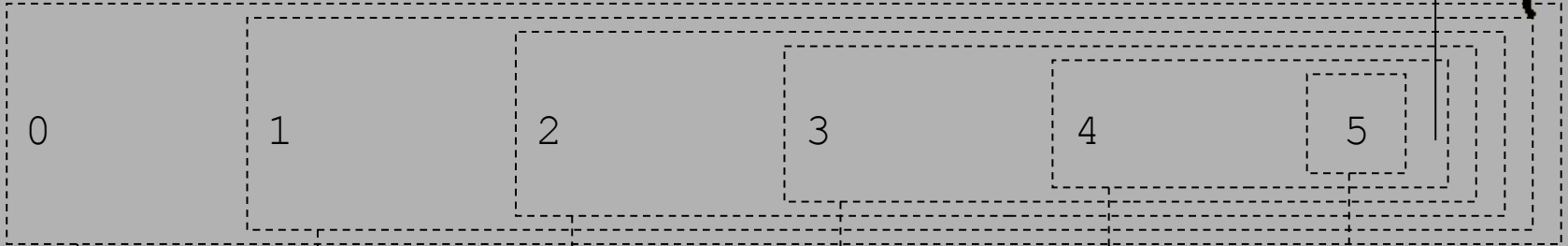


3

```
/* εκτύπωση τιμών από 0 μέχρι n */  
  
#include <stdio.h>  
  
void printInts(int from, int to) {  
    int i;  
  
    for(i=from; i <= to; i++) {  
        printf("%d ", i);  
    }  
}  
  
int main(int argc, char *argv[]) {  
    int n;  
  
    printf("enter int: ");  
    scanf("%d", &n);  
    printInts(0, n);  
  
    return(0);  
}
```



4



τύπωση 0
τύπωση 1-5

τύπωση 1
τύπωση 2-5

τύπωση 2
τύπωση 3-5

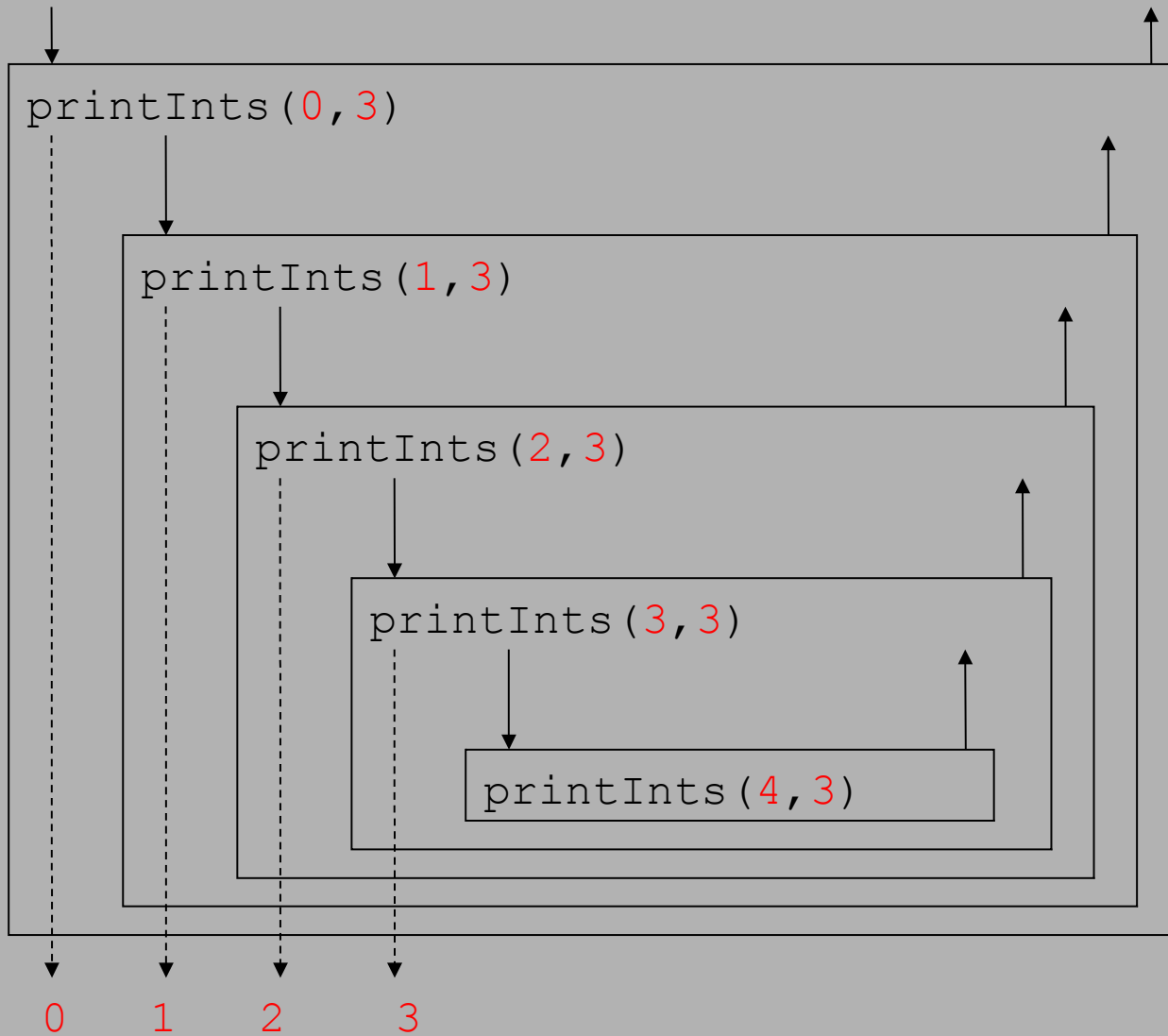
τύπωση 3
τύπωση 4-5

τύπωση 4
τύπωση 5-5

τύπωση 5
τύπωση 6-5



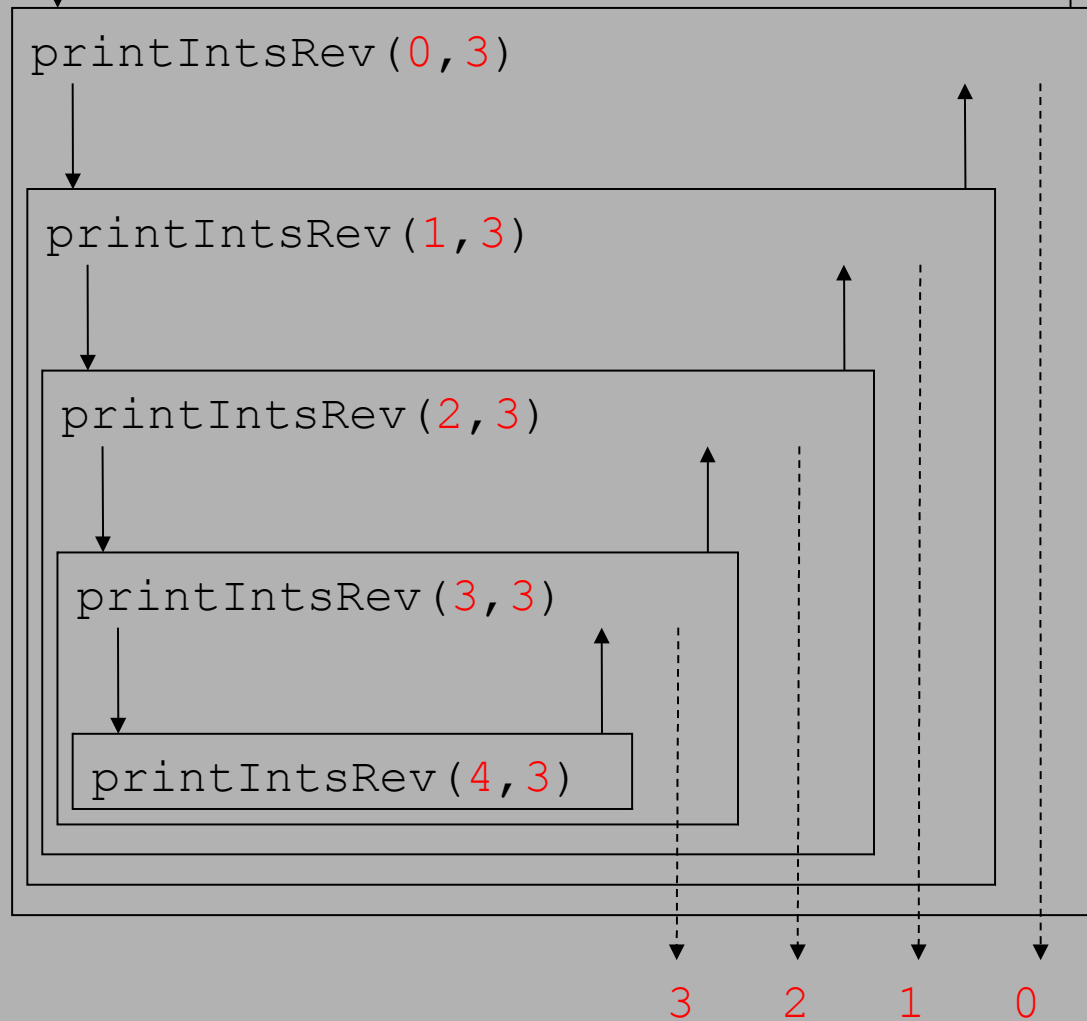
```
/* εκτύπωση τιμών από 0 μέχρι n */  
  
#include <stdio.h>  
  
void printInts(int from, int to) {  
    if (from <= to) {  
        printf("%d ", from);  
        printInts(from+1, to);  
    }  
}  
  
int main(int argc, char *argv[]) {  
    int n;  
  
    printf("enter int: ");  
    scanf("%d", &n);  
    printInts(0, n);  
  
    return(0);  
}
```



Εκτύπωση τιμών από n έως 0



7





```
/* εκτύπωση τιμών από n μέχρι 0 */  
  
#include <stdio.h>  
  
void printIntsRev(int from, int to) {  
    if (from<=to) {  
        printIntsRev(from+1, to);  
        printf("%d ", from);  
    }  
}  
  
int main(int argc, char *argv[]) {  
    int n;  
  
    printf("enter int: ");  
    scanf("%d", &n);  
    printIntsRev(0, n);  
  
    return(0);  
}
```


Αναδρομική σκέψη και λύση προβλημάτων



9

- Για να λυθεί ένα πρόβλημα με αναδρομή, πρέπει πρώτα να **εκφραστεί** με αναδρομικό τρόπο.
- Κλασική προσέγγιση:
 1. Προσπαθούμε να βρούμε την λύση του προβλήματος για την πιο απλή περίπτωση του.
 2. Στην συνέχεια, ανάγουμε/κατασκευάζουμε την λύση της (αμέσως) πιο πολύπλοκης περίπτωσης, με βάση την λύση της πιο απλής περίπτωσης (αναδρομή).
 3. Αν τα (1) και (2) γίνουν «σωστά», έχουμε **ήδη** κατασκευάσει την λύση στο πρόβλημα μας!
- Η συνθήκη τερματισμού είναι πολλές φορές το κλειδί στην ανεύρεση της πιο απλής περίπτωσης.

«Συμβατικός» υπολογισμός x!



10

```
/* υπολογισμός n! */  
  
int factorial(int n) {  
    int i, res;  
  
    res = 1;  
    for (i=2; i<=n;i++)  
        res = res * i;  
  
    return(res);  
}
```



Αναδρομικός υπολογισμός $x!$

- Επιθυμούμε να υπολογίζουμε την έκφραση

$$x! == 1 * 2 * 3 * \dots * (x-1) * x$$

- Περίπτωση τερματισμού

$x == 0$ ή $x == 1$: επιστρέφεται 1

- Γενική περίπτωση

$x > 1$: επιστρέφεται $x * (x-1)!$

- Η «λύση» της γενικής περίπτωσης μπορεί να κατασκευαστεί με βάση την λύση του **ίδιου** προβλήματος, σε **μικρότερη** «κλίμακα».



```
/* υπολογισμός n! */  
  
int factorial(int n) {  
  
    if (n<=1) {  
        return(1);  
    }  
    else {  
        return(n*factorial(n-1));  
    }  
  
}
```

