# Προγραμματισμός Ι (ΗΥ120)

## Διάλεξη 16:
## Εισαγωγή στην Αναδρομή

**Χρήστος Δ. Αντωνόπουλος**
**13/12/2016**

**Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών & Δικτύων**
**Πανεπιστήμιο Θεσσαλίας**

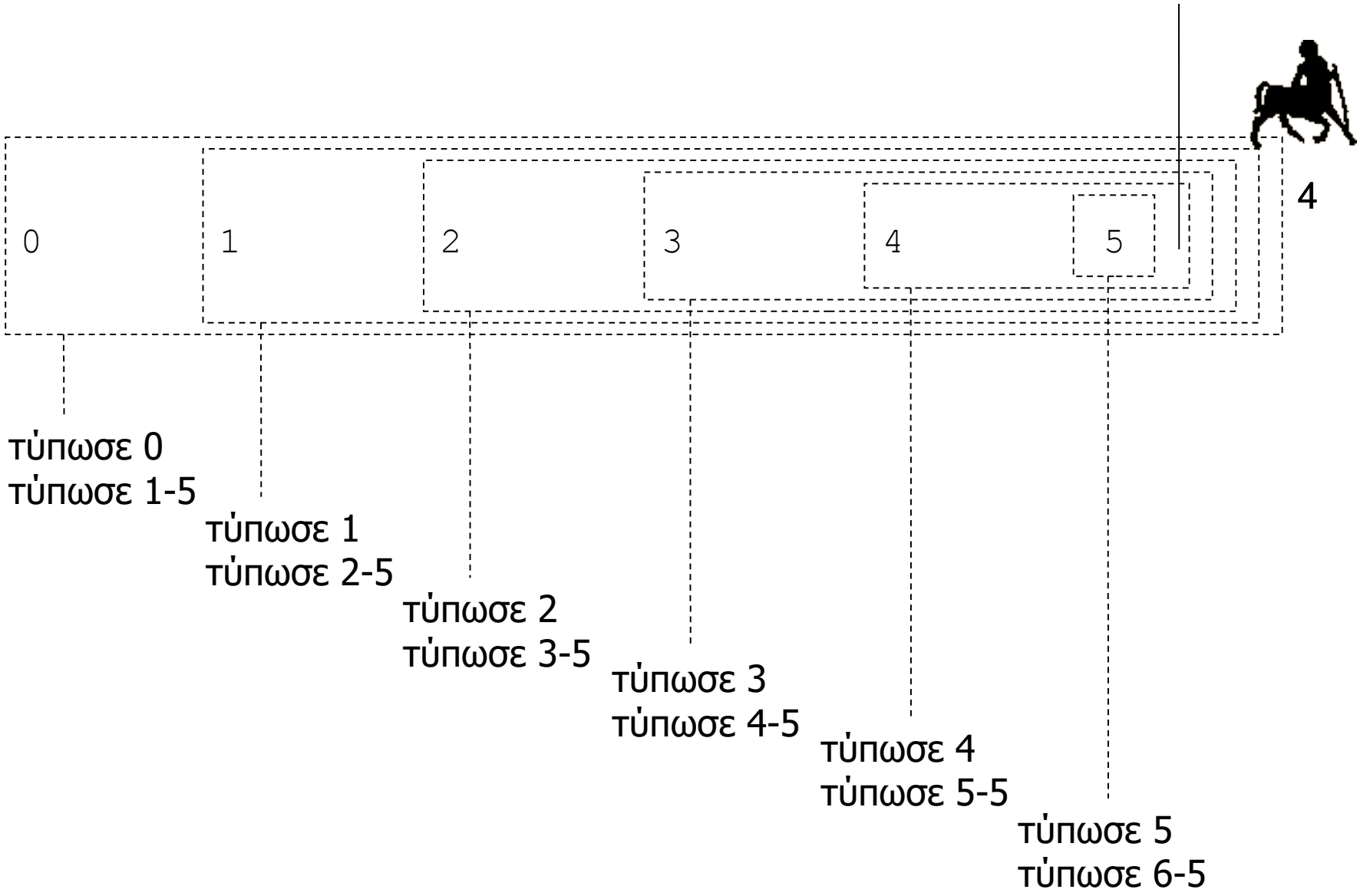
# Αναδρομικές Συναρτήσεις



- 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.

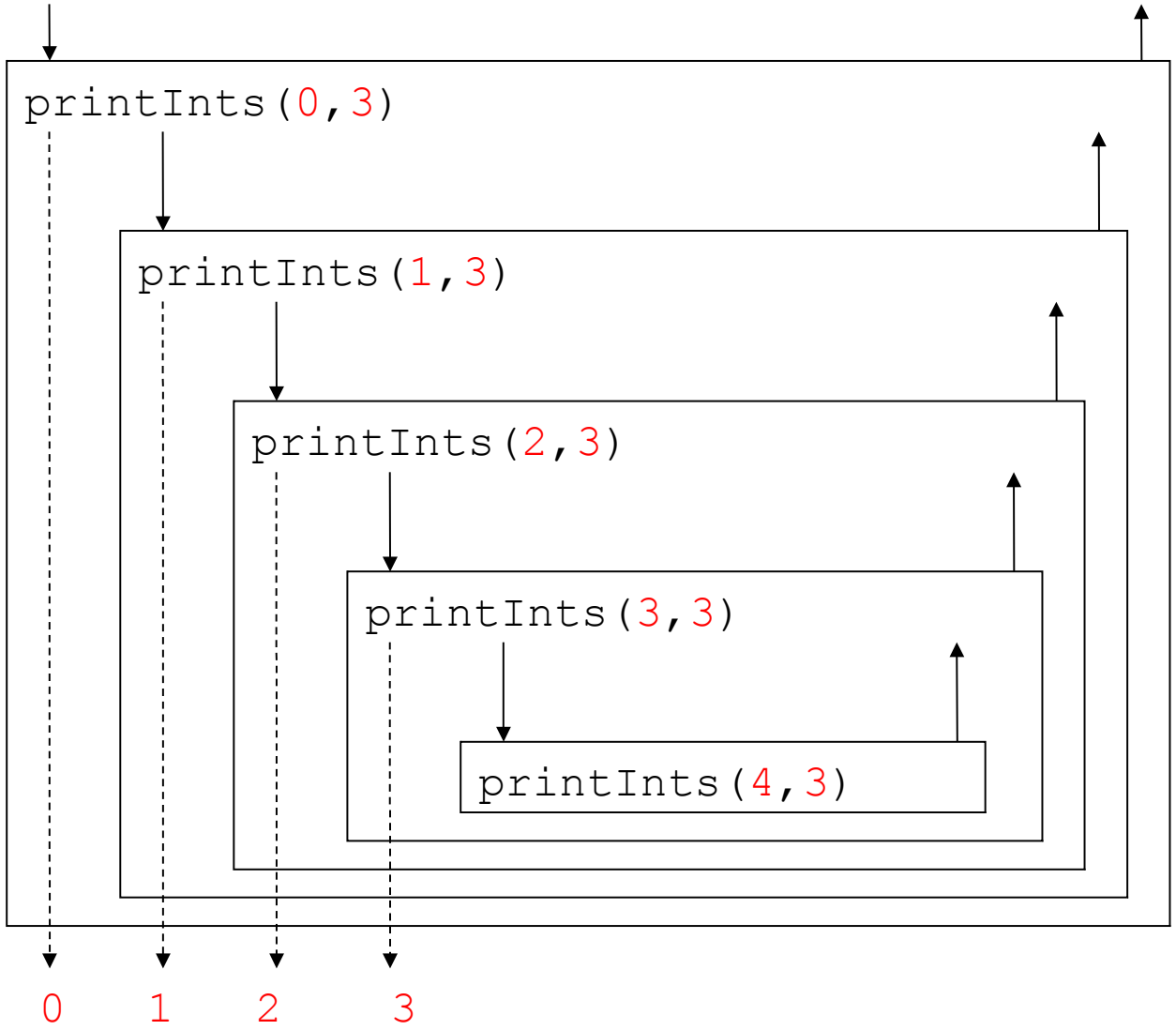


```
/* εκτύπωση τιμών από 0 μέχρι n */

#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);
}
```

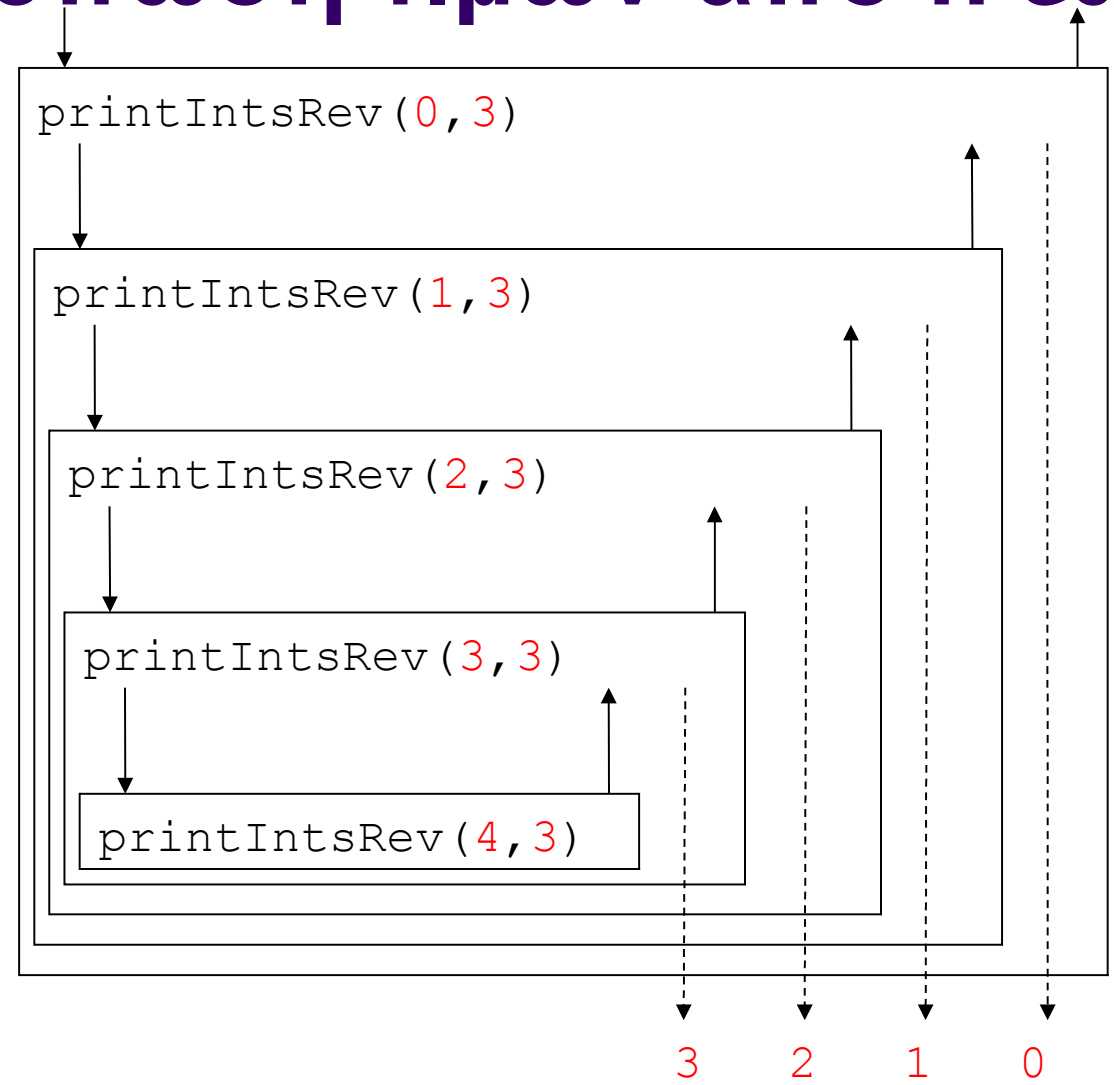

0
1
2
3
4
5
4
τύπωσε 0
τύπωσε 1-5
τύπωσε 1
τύπωσε 2-5
τύπωσε 2
τύπωσε 3-5
τύπωσε 3
τύπωσε 4-5
τύπωσε 4
τύπωσε 5-5
τύπωσε 5
τύπωσε 6-5



```
/* 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);
}
```


printInts(0,3)
printInts(1,3)
printInts(2,3)
printInts(3,3)
printInts(4,3)
0
1
2
3

# Εκτύπωση τιμών από n έως 0





```
/* 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



- 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!



```
/* 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*…*(x-1)*x`

- Περίπτωση τερματισμού

  `x==0` ή `x==1`: επιστρέφεται `1`

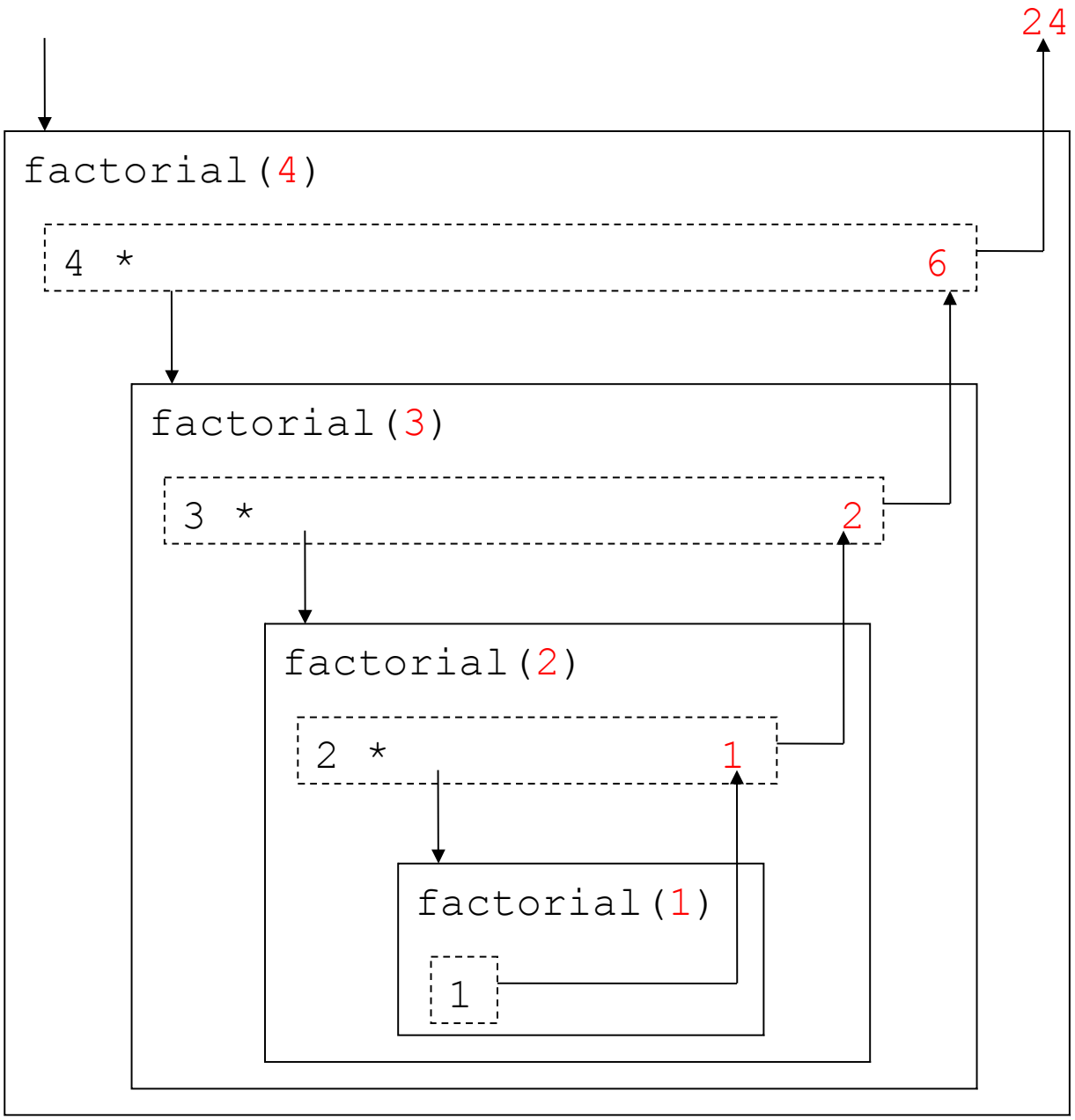
- Γενική περίπτωση

  `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));
  }

}
```


24
factorial(4)
4 *
6
factorial(3)
3 *
2
factorial(2)
2 *
1
factorial(1)
1