

# ΑΝΑΔΡΟΜΗ (Recursion)

- Αναδρομικός Ορισμός του  $n!$

$$0! = 1$$

$$n! = n (n-1)! , n > 0$$

- Αναδρομικός Ορισμός Συνάρτησης

- Βάση Αναδρομής

- Αναδρομικό Βήμα

- Αναδρομικός Υπολογισμός του  $3!$

$$3! = 3 * 2!$$

$$2! = 2 * 1!$$

$$1! = 1 * 0!$$

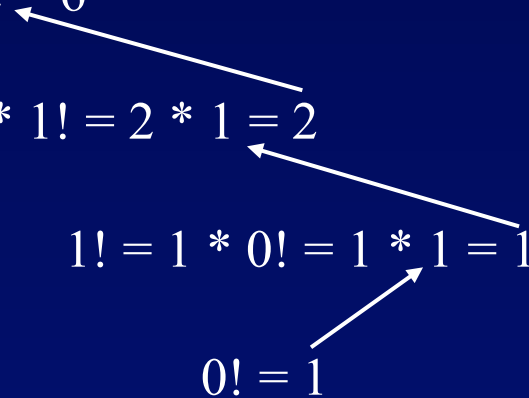
$$0! = 1$$

$$3! = 3 * 2! = 3 * 2 = 6$$

$$2! = 2 * 1! = 2 * 1 = 2$$

$$1! = 1 * 0! = 1 * 1 = 1$$

$$0! = 1$$



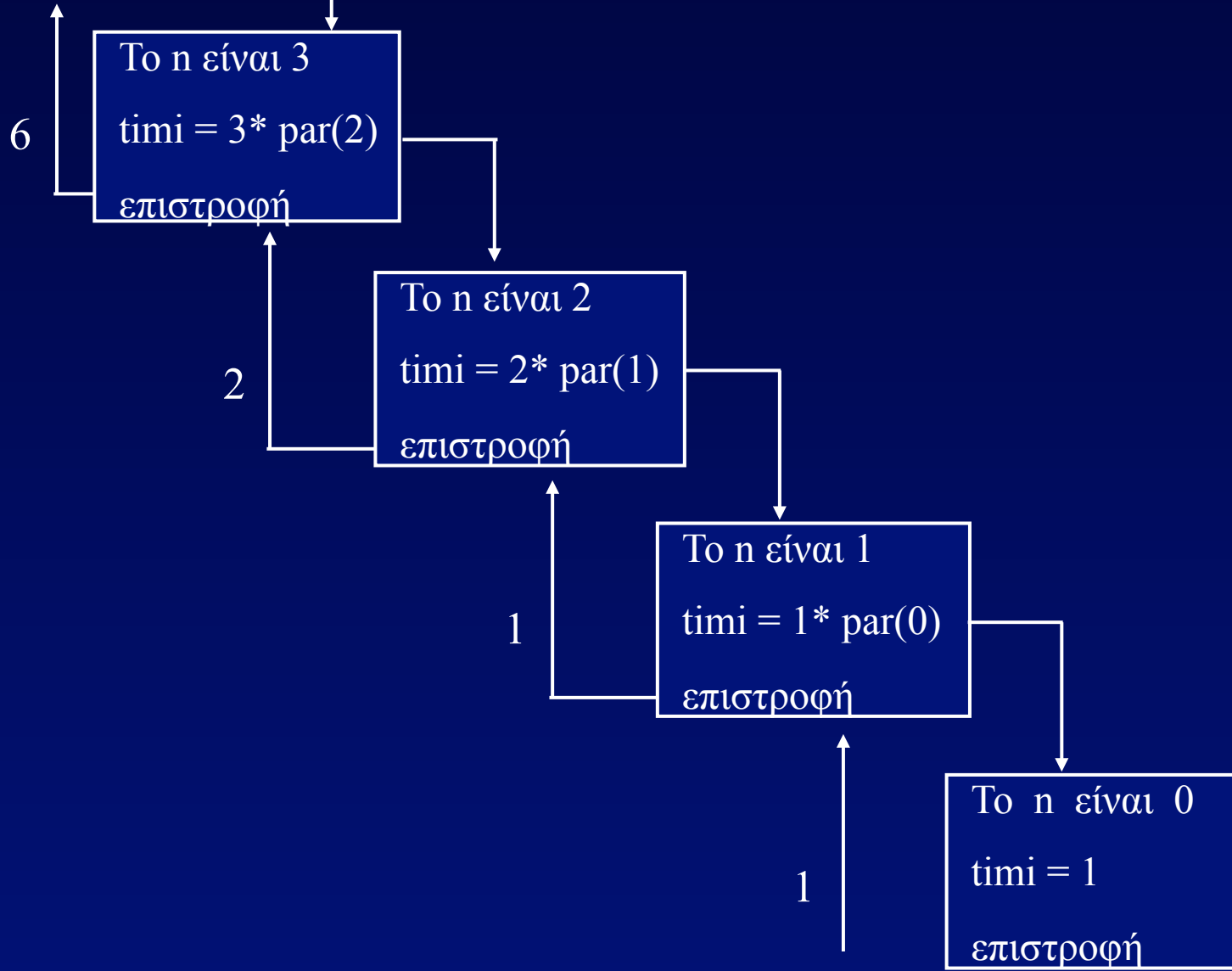
Ο αναδρομικός αυτός ορισμός μπορεί να υλοποιηθεί εύκολα όπως φαίνεται στο παρακάτω υποπρόγραμμα:

```
int par(int n)
{
    int    timi;
    if (n == 0) /* βήμα διακοπής */
        timi = 1;
    else /* αναδρομικό βήμα    $n! = n * (n-1)!$  */
        timi = n * par(n - 1);
    return (timi);
}

/* A */
```

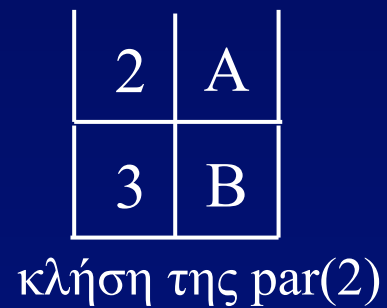
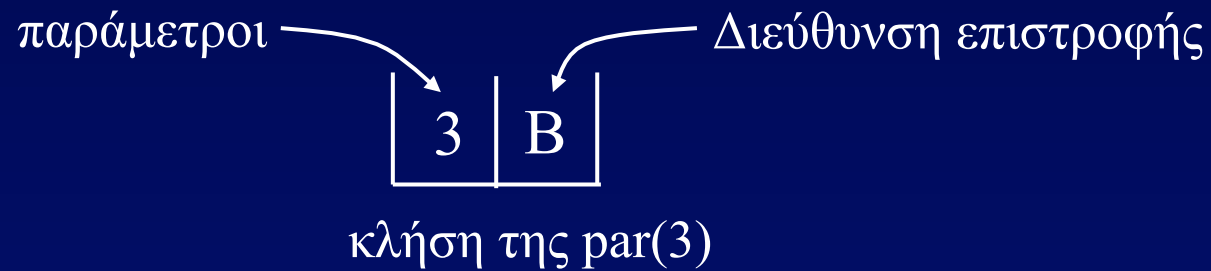
Στο ακόλουθο σχήμα παρουσιάζεται ο τρόπος λειτουργίας του αναδρομικού υποπρογράμματος, όπου κάθε πλαίσιο σχετίζεται με αυτό που ονομάζεται **εγγραφή ενεργοποίησης (activation record)** για το υποπρόγραμμα.

paragontiko = par(3);

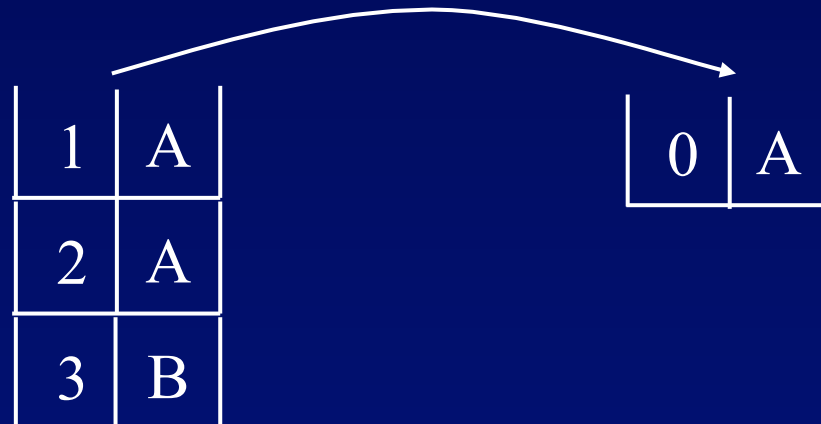


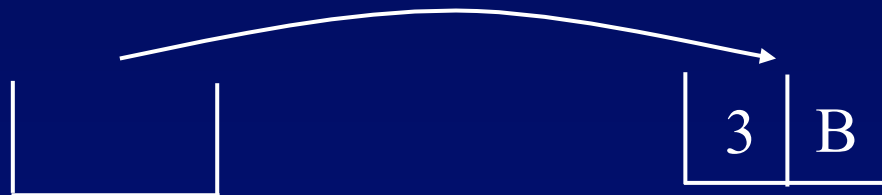
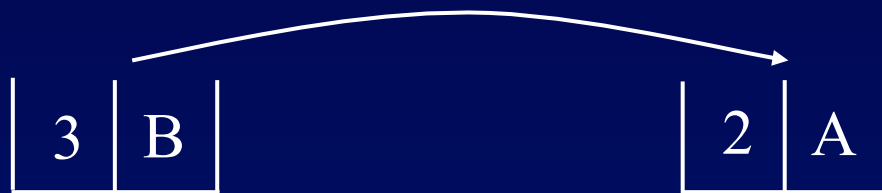
## Υλοποίηση Αναδρομής

```
{ /* κύριο πρόγραμμα */  
{B} x = par(3);  
} /* κύριο πρόγραμμα */
```



0	A
1	A
2	A
3	B







## Πολυπλοκότητα

$$T(n) = 2 + T(n - 1) = 2 + 2 + T(n - 2) = 2 + 2 + \dots + 2 + T(0) = 2n + 1$$

$$T_A(n) = O(n)$$

$$n! = 1 * 2 * 3 * \dots * n, \quad T_E(n) = O(n)$$

## Αριθμοί Fibonacci

0 1 1 2 3 5 8 13 21 ...

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

## Πολυπλοκότητα

$$\begin{aligned} T(n) &= 3 + T(n-1) + T(n-2) = 3 + 3 + T(n-2) + T(n-3) + T(n-2) \\ &= 6 + 2T(n-2) + T(n-3), \text{ για } n \geq 3 \end{aligned}$$

$$\begin{aligned} T(n) &\geq 2T(n-2) \geq 2^2 T(n-4) \geq \dots \geq 2^{n/2} T(0), \text{ αν } n \text{ άρτιο} \\ \text{ή} \\ T(n) &\geq 2T(n-2) \geq 2^2 T(n-4) \geq \dots \geq 2^{(n-1)/2} T(1), \text{ αν } n \text{ περιττό} \end{aligned}$$

Αλλά  $T(0) = T(1) = 1$ , συνεπώς

$$T_A(n) = \Omega(2^{n/2}) \quad \text{για όλα τα } n \geq 2$$

```
int eFib (int n)
```

```
/* Επαναληπτικό υποπρόγραμμα για τον υπολογισμό του n-οστού αριθμού  
Fibonacci για n >=3 */
```

```
{  
    int    Fib1, Fib2, Fib3, i;  
  
    Fib1 = 1;  
    Fib2 = 1;  
    for (i = 3; i<=n; i++)  
    {  
        Fib3 = Fib1 + Fib2;  
        Fib1 = Fib2;  
        Fib2 = Fib3;  
    }  
    return(Fib3);  
}
```

$T_E = O(n)$

## Μια περίπτωση αφαίρεσης της αναδρομής

```
void grammiki_anadromi (int n)
{
    if (συνθήκη (n))
        βάση αναδρομής;
    else
    {
        προηγούμενες πράξεις (n);
        grammiki_anadromi (F(n));
        μετέπειτα πράξεις (n);
    }
}
```

```
void epanaliptiki (int n)
{
    typos_stoivas stoiva;

    dimiourgia(stoiva);
    while (!συνθήκη (n))
    {
        προηγούμενες πράξεις (n);
        othisi (stoiva, n);
        n = F(n);
    }
    βάση αναδρομής;
    while (!keni(stoiva))
    {
        exagogi(stoiva, n);
        μετέπειτα πράξεις (n);
    }
}
```