

Κεφάλαιο

15

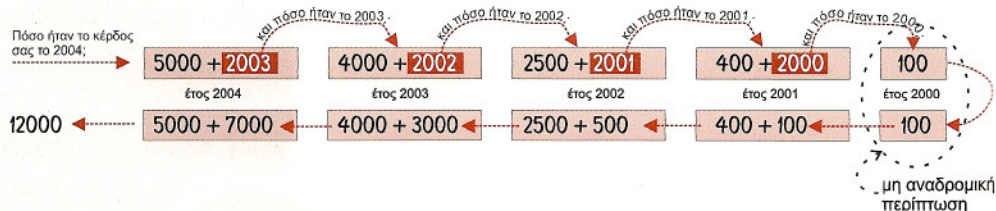
Αναδρομή




Αναδρομή

Αναδρομή είναι η διαδικασία κατά την οποία ορίζουμε μια λειτουργία σε σχέση με την ίδια τη λειτουργία. Λίγο δυσνόητο αλλά ας δώσουμε ένα απλό παράδειγμα:

Αν ρωτήσουμε τον διευθυντή μιας εταιρείας "Πόσα είναι τα κέρδη της εταιρείας σας το 2004" μπορεί να μας απαντήσει ως εξής: "είναι 5.000€ περισσότερα από το 2003". Εμείς φυσικά θα ρωτήσουμε "και πόσα ήταν το 2003" και θα λάβουμε παρόμοια απάντηση, π.χ. "4.000€ περισσότερα από ό,τι το 2002" κ.ο.κ. Αυτό θα επαναλαμβάνεται συνέχεια, όχι όμως επάπειρο επειδή κάποια στιγμή ο διευθυντής δε θα μπορεί να μας απαντήσει με τον ίδιο τρόπο αλλά θα πρέπει να μας πει ότι είχανε συγκεκριμένα κέρδη (π.χ. 100€) διότι θα έχουμε φτάσει πια στη χρονιά που ιδρύθηκε η εταιρεία. Τότε εμείς (που σημειώναμε τις ερωτήσεις που κάναμε και τις απαντήσεις που παίρναμε) θα μπορέσουμε, ξεκινώντας από την τελευταία απάντηση που πήραμε (τα 100€) και προχωρώντας προς τα εμπρός, να υπολογίσουμε τα κέρδη του 2004 που αρχικά ζητήσαμε.



Στο παραπάνω σχήμα υπάρχει μία περίπτωση στην οποία ο διευθυντής δεν μας παραπέμπει στα κέρδη του προηγούμενου έτους αλλά μας απαντάει συγκεκριμένα. Είναι η τελευταία ερώτηση, που αφορά στο έτος 2000 και μας απαντάει συγκεκριμένα ότι το κέρδος ήταν 100€. Έτσι ξεκαθαρίζουν όλα και γυρνώντας βήμα-βήμα προς τα εμπρός μπορούμε να απαντήσουμε στην αρχική ερώτηση.

 Κάθε αναδρομική διαδικασία πρέπει να έχει και μία **μη αναδρομική περίπτωση**, διαφορετικά θα συνεχίζεται επ' άπειρο.

Στη C μια συνάρτηση μπορεί να καλεί τον εαυτό της. Αν στο σώμα μιας συνάρτησης μια πρόταση καλεί την ίδια τη συνάρτηση τότε η συνάρτηση λέγεται **αναδρομική** (recursive).

Ας θεωρήσουμε την επόμενη συνάρτηση, η οποία επιστρέφει ως τιμή το παραγοντικό ενός αριθμού n ($1*2*3 \dots * n$). Για παράδειγμα, το παραγοντικό του 3 είναι 6 ($1*2*3$) και του 8 είναι 40320 ($1*2*3*4*5*6*7*8$).

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

Η προηγούμενη συνάρτηση υπολογίζει (με τον κλασικό τρόπο) και επιστρέφει ως τιμή το παραγοντικό ενός αριθμού n .

Το παραγοντικό ($n!$) ενός αριθμού n είναι το γινόμενο όλων των ακέραιων αριθμών από το 1 μέχρι το n . Επίσης, το παραγοντικό ($n!$) ενός αριθμού n ορίζεται ως το γινόμενο του n και του παραγοντικού του προηγούμενου αριθμού δηλαδή $n! = n * (n-1)!$, δηλαδή το παραγοντικό του 6 είναι $6 * 5!$. Ο υπολογισμός λοιπόν του παραγοντικού μπορεί να γίνει με αναδρομική διαδικασία.

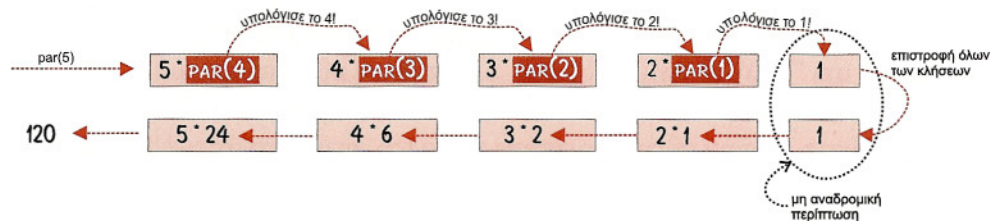
```
par(n)
int n;
{
    int p;
    if(n==1) return 1;
    p=n*par(n-1);
    return p;
}
```

Έλεγχος για μη αναδρομική περίπτωση

Εάν η παραπάνω συνάρτηση κληθεί π.χ. σε μια παράσταση με παράμετρο 5,

`a=par(5);`

θα ακολουθήσει η επόμενη διαδικασία με την οποία τελικά η συνάρτηση θα επιστρέψει την τιμή της, η οποία θα αποθηκευτεί στη μεταβλητή `a`:



Οι αναδρομικές κλήσεις της συνάρτησης `par()` θα σταματήσουν όταν ζητήσει το παραγοντικό του 1, που αποτελεί τη μη αναδρομική περίπτωση της όλης διαδικασίας.

Παραδείγματα

Π.1 Η επόμενη αναδρομική συνάρτηση δέχεται ως παράμετρο έναν αριθμό και τον εμφανίζει αντίστροφα. Για παράδειγμα, αν δοθεί ο 4672 εμφανίζει τον 2764.

```
int reverse(int n)
{
    if (n==0)
        return;
    else
    {
        printf("%d", n%10);
        return reverse(n/10);
    }
}
```

Μη αναδρομική περίπτωση

Εμφανίζει το υπόλοιπο της διαίρεσης με το 10, δηλαδή το λιγότερο σημαντικό ψηφίο του αριθμού.

Καλεί αναδρομικά τη συνάρτηση με παράμετρο το πηλίκο του αριθμού διά του 10.

Π.2 Ο επόμενος τύπος, που προέρχεται από τον Leonard Euler(1707-1783), υπολογίζει προσεγγιστικά το π . Το πρόγραμμα που ακολουθεί χρησιμοποιεί αυτή τη σχέση για να υπολογίσει την τιμή του π . Το πρόγραμμα χρησιμοποιεί αναδρομική συνάρτηση για τον υπολογισμό του δεξιού μέλους της σχέσης. Η συνάρτηση δέχεται ως παράμετρο το n . Χρησιμοποιούνται οι 1000 πρώτοι όροι.

$$\frac{\pi^2}{6} = \sum_{n=1}^{\infty} \frac{1}{n^2}$$

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
double melos();
```

```
main()
```

```
{
    double p;
    p=sqrt(melos(1000)*6);
    printf("p=%f\n",p);
}
```

Επιστρέφει την τιμή του δεξιού μέλους της σχέσης για τους 1000 πρώτους όρους.

Υπολογίζει την τιμή του p (π) επιλύοντας τη σχέση ως προς π .

```
double melos(int n)
```

```
{
    double res;
    if(n==1)
        return 1.0;
    else
    {
        res=1.0/(n*n)+melos(n-1);
        return res;
    }
}
```

Μη αναδρομική περίπτωση.

Ανασκόπηση Κεφαλαίου 15

- Αναδρομική είναι μια διαδικασία η οποία μπορεί να οριστεί σε σχέση με την ίδια τη διαδικασία.
- Αναδρομική συνάρτηση είναι η συνάρτηση η οποία καλεί ξανά τον εαυτό της.
- Κάθε αναδρομική συνάρτηση έχει τουλάχιστον μία μη αναδρομική περίπτωση.

Ασκήσεις Κεφαλαίου 15

15.1 Να γραφεί αναδρομική συνάρτηση η οποία να υπολογίζει το άθροισμα της σειράς $1/1 + 1/2 + 1/3 + 1/4 + \dots + 1/n$. Η συνάρτηση θα δέχεται ως παράμετρο τον αριθμό n . ★ ★

15.2 Να εντοπιστεί το λάθος στην επόμενη αναδρομική συνάρτηση. ★

```
par(int n)
{
    int p;
    p=n+par(n-1);
    return p;
}
```

15.3 Τι κάνει η επόμενη συνάρτηση; ★

```
par(int n)
{
    int p;
    if (n==1) return 0;
    p=n+par(n/2);
    return p;
}
```

Τι τιμή θα επιστρέψει η `par(20)`;

- 15.4** Ο επόμενος τύπος υπολογίζει προσεγγιστικά το π . Να γραφεί πρόγραμμα το οποίο να χρησιμοποιεί αυτό τον τύπο και να υπολογίζει την τιμή του π . Για τον υπολογισμό του δεξιού μέλους να χρησιμοποιηθούν αναδρομικές συναρτήσεις. Να χρησιμοποιηθούν οι 1000 πρώτοι όροι. ★ ★ ★

$$\pi = 2 \times \frac{2}{\sqrt{2}} \times \frac{2}{\sqrt{2+\sqrt{2}}} \times \frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}} \times \dots$$

- 15.5** Ποια από τα επόμενα αληθεύουν: ★

- ☐ Κάθε αναδρομική συνάρτηση πρέπει να έχει μία τουλάχιστον μη αναδρομική περίπτωση.
- ☐ Όλες οι συναρτήσεις μπορούν να γραφούν με αναδρομική μορφή.
- ☐ Μια αναδρομική συνάρτηση μπορεί να επιφέρει εξάντληση της μνήμης του Η/Υ.
- ☐ Όλες οι γλώσσες προγραμματισμού υποστηρίζουν αναδρομικές συναρτήσεις.
- ☐ Μια αναδρομική συνάρτηση πρέπει να έχει τουλάχιστον μία παράμετρο.