



## Κεφάλαιο 8.7

# Πίνακες & Συναρτήσεις (Διάλεξη 17)

Διδάσκων: Δημήτρης Ζεϊναλιπούρ



# Εισαγωγή

- Στις προηγούμενες διαλέξεις μάθαμε πώς να **δηλώνουμε, αρχικοποιούμε** και να **επεξεργαζόμαστε** πίνακες.
- Σήμερα θα μελετήσουμε πώς μπορούμε να **περάσουμε** ένα **πίνακα** σε μια **συνάρτηση**.
- Αυτό είναι χρήσιμο διότι μας επιτρέπει να επωφεληθούμε όλων των πλεονεκτημάτων των συναρτήσεων όταν χρησιμοποιούμε πίνακες.

Σήμερα θα δούμε:

- 1) Κλήση με Τιμή
- 2) Κλήση με Αναφορά
- 3) Πίνακες και Συναρτήσεις
- 4) Παραδείγματα



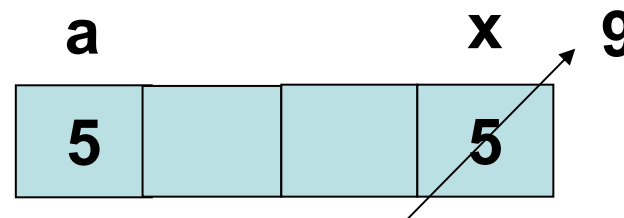
# 1) Κλήση με Τιμή

- Θυμηθείτε ότι όταν περάσουμε μια μεταβλητή (π.χ. int, char, float), σε μια συνάρτηση τότε δημιουργείτε ένα αντίτυπο της μεταβλητής, το οποίο δεν έχει καμία σχέση με την αρχική μεταβλητή. Αυτός ο τρόπος περάσματος παραμέτρων ονομάζεται :

**Κλήση με τιμή (call by value)**

```
int main() {  
    int a = 5;  
    printf("Before:%d, After:%d", a, add(a));  
}
```

```
function add(int x) {  
    return x+4;  
}
```





# 1) Πέρασμα Παραμέτρου Δια Τιμής

- Η συνάρτηση έχει μία **τοπική μεταβλητή** για να αποθηκεύσει το δικό της αντίγραφο της τιμής που της δίνουμε
- Όταν κάνουμε αλλαγές στο **αντίγραφο**, η **αρχική τιμή παραμένει η ίδια**
- Αν θέλουμε να επιστρέψουμε τιμές πρέπει να χρησιμοποιήσουμε τη **return**
- Αυτό λέγεται **κλήση με τιμή** (call by value)



## 2) Κλήση με Αναφορά

- Τι γίνεται αν θέλουμε να περάσουμε ένα πίνακα σε μια συνάρτηση?
- Αν ο πίνακας περνούσε στην συνάρτηση με τον ίδιο τρόπο που περνά η μεταβλητή (δηλαδή με copy), τότε κάθε κάλεσμα θα σήμαινε την αντιγραφή ενός μεγάλου αριθμού στοιχείων  
**=> Αυτό θα ήταν Πολύ ακριβό.**
- **Αντί** να δημιουργείτε ένα **νέο αντίγραφο**, στην γλώσσα C όταν περνάμε ένα πίνακα σε μια συνάρτηση τότε περνά στην συνάρτηση μόνο η κάποια **αναφορά στον πίνακα**. Αυτό το κάλεσμα ονομάζεται :

### Κλήση με Αναφορά (call by reference)

Δηλαδή δεν γίνεται αντιγραφή των στοιχείων στην συνάρτηση

Αυτό σημαίνει ότι μπορούμε να αλλάζουμε τους πίνακες μέσα στις συναρτήσεις



## 2) Κλήση με Αναφορά

- Η κλήση με αναφορά δεν είναι κάτι νέο.
- Μπορεί να γίνει και με κανονικές μεταβλητές.

π.χ. δείτε το πιο κάτω παράδειγμα

### Κλήση με Τιμή

```
int change(int a) {  
    a = 5;  
}
```

```
int main( ) {  
    int a = 1;  
    change(a);  
    printf("%d", a);  
}
```

ΕΚΤΥΠΩΝΕΙ «1»

### Κλήση με Αναφορά για Μεταβλητές

```
int change(int *a) {  
    (*a) = 5;  
}
```

```
int main( ) {  
    int a = 1;  
    change(&a);  
    printf("%d", a);  
}
```

ΕΚΤΥΠΩΝΕΙ «5»

Δεν  
χρειάζεται  
για το  
μάθημα

Δηλαδή μπορούμε να αλλάξουμε το a χωρίς return 17-6



### 3) Πίνακες και Συναρτήσεις

Χρησιμοποιούμε τις ακόλουθη σύνταξη όταν περνούμε πίνακες σε συναρτήσεις:

- Στο πρότυπο και τον ορισμό της συνάρτησης :  
**void FillArray ( int array[ ], int size);**  
ή  
**void FillArray ( int [ ], int);**
- Στην κλήση της συνάρτησης:  
**FillArray ( array, size);**

# 3) Πίνακες και Συναρτήσεις (Παραδειγμα)



```
#include <stdio.h>
#define SIZE 4
// Πρότυπο
void FillArray (int[ ], int) ;

main ( ) {
    int array [SIZE];

    // Κάλεισμα Συνάρτησης
    FillArray ( array, SIZE );
}
```

```
// Ορισμός Συνάρτησης
void FillArray(int array[ ], int L)
{
    int i;
    for ( i = 0; i < L; i++) {
        array [i] = i;
    }
}
```

Κατακρίβειαν το SIZE μπορούσε να χρησιμοποιηθεί κατευθείαν εδώ (μιας και είναι καθολική μεταβλητή)

Έξοδος

array[0] = 0

array[1] = 1

array[2] = 2

array[3] = 3





# Παράδειγμα 1

Γράψετε μια συνάρτηση η οποία αρχικοποιεί όλες τις θέσεις ενός πίνακα ακεραίων `ARRAY` με μέγεθος `SIZE`, στην τιμή `0`



# Παράδειγμα 1

```
#include <stdio.h>
```

```
void InitArray (int array [ ], int size)
{
    int i;
    for ( i = 0; i < size; i++ )
    {
        array [ i ] = 0 ;
    }
}
```

```
main ( ) {
    int array[] = {1,2,3,4,5,6};
    InitArray (array, 6);
}
```



## Παράδειγμα 2

Γράψετε μια **συνάρτηση** η οποία λαμβάνει ως τιμή εισόδου ένα **πίνακα ακεραίων**, και το **μέγεθος του πίνακα**, και επιστρέφει σαν τιμή εξόδου τον **μέσο όρο**



# Παράδειγμα 2

```
#include <stdio.h>
```

```
float avg(int [ ], int );
```

```
main ( ) {
```

```
    int array[]= {1,2,3,4,5,6};
```

```
    printf("Average: %f", avg(array, 6));
```

```
}
```

```
float avg(int array[ ], int L) {
```

```
    int i; int sum=0, count=0;
```

```
    for (i = 0; i < L; i++) {
```

```
        sum += array[i];
```

```
        count++;
```

```
    }
```

```
    return (float)sum/count;
```

```
}
```



## Παράδειγμα 3

Γράψετε μια συνάρτηση η οποία επιστρέφει το άθροισμα των αριθμών μεταξύ των θέσεων **[2..4]** ενός πίνακα ακεραίων.

Αν ο πίνακας περιέχει λιγότερα από **5 στοιχεία** τότε εκτυπώνετε μήνυμα λάθους και επιστρέφει -1 η συνάρτηση

### Σημείωση

Η πρώτη θέση του πίνακα είναι 0.

Π.χ. `int array[] = {1,2,3,4,5,6};` επιστρέφει **12**

Π.χ. `int array[] = {1,2,3};` επιστρέφει **-1**



## Παράδειγμα 3

```
int selective_sum(int array[ ], int L) {  
    int i;  
    int sum=0;  
  
    if (L<5) {  
        printf("Error"); return -1;  
    }  
  
    for (i = 2; i <5; i++) {  
        sum += array[i];  
    }  
    return sum;  
}
```

Π.χ. int array[] = {1, 2, 3, 4, 5, 6};

επιστρέφει 12



## Παράδειγμα 4

Γράψετε μια συνάρτηση η οποία επιστρέφει το άθροισμα των **περιττών θέσεων** ενός πίνακα ακεραίων.

### Σημείωση

Η πρώτη θέση του πίνακα είναι 0. Επομένως θέλουμε το άθροισμα των στοιχείων στις θέσεις 1, 3, 5, ...



# Παράδειγμα 4

## Τρόπος Α'

```
int oddsum(int array[ ], int L) {  
    int i;  
    int sum=0;  
  
    for (i = 0; i < L; i++) {  
        if (i%2 == 1) {  
            sum += array[i];  
        }  
    }  
  
    return sum;  
}
```

Π.χ. int array[] = {1, 2, 3, 4, 5, 6};

## Τρόπος Β'

```
int oddsum(int array[ ], int L) {  
    int i;  
    int sum=0;  
  
    for (i = 1; i < L; i+=2) {  
        sum += array[i];  
    }  
  
    return sum;  
}
```

επιστρέφει 12





## Παράδειγμα 5

Γράψετε μια συνάρτηση η οποία παίρνει τις εξής παραμέτρους:

Κάποιο πίνακα ακεραίων ARRAY,

Το μέγεθος του πίνακα SIZE,

Και στην συνέχεια ζητά από τον χρήστη να δώσει SIZE θετικούς ακεραίους  $>0$ , οι οποίοι αποθηκεύονται στον πίνακα ARRAY.



# Παράδειγμα 5

```
void getValues (int array [ ], int SIZE)
{
    int i;
    for ( i = 0 ; i < SIZE ; i++)
    {
        printf ("Enter next value : ") ;
        scanf ("%d", &array[i] ) ;
        while ( array [ i ] < 1)
        {
            printf ("Values must be positive...");
            printf ("Enter next value : ") ;
            scanf ("%d", &array[ i ] ) ;
        }
    }
}
```



# Παράδειγμα 6 – Σκελετός Προγράμματος

```
#include <stdio.h>
#define SIZE      39
#define GRADES    6

// Πρότυπα Συναρτήσεων
void    PrintInstructions (void) ;
void    InitArray (int array [ ], int size) ;
void    FillArray (int array [ ], int size) ;
double  ProcessGrades (int score [ ], int size, int gradeCount [ ] ) ;
double  FindAverage (double sum, int num) ;
void    PrintResults (double average, int gradeCount [ ] ) ;
```



# Παράδειγμα 6 – Σκελετός Προγράμματος

```
int main ( )
{
    int i, score [SIZE], gradeCount [GRADES] ;
    double average;

    PrintInstructions ( ) ;                // Εκτύπωση Οδηγιών
    InitArray (gradeCount, GRADES) ;      // Αρχικοποίηση πίνακα gradeCount σε 0
    FillArray (score, SIZE) ;             // Εισαγωγή στοιχείων από χρήστη

    // εύρεση μέσου όρου, αλλαγή θέσεων gradeCount από πίνακα score
    average = ProcessGrades (score, SIZE, gradeCount ) ;

    // Εκτύπωση αποτελεσμάτων gradeCount
    PrintResults (average, gradeCount) ;
    return 0;
}
```