## Μηχανισμός Call By reference

Η C είναι τυπικά call by value, αλλά αν περάσεις σε μια συνάρτηση τη διεύθυνση μιας μεταβλητής (pointer), η συνάρτηση μπορεί να αλλάξει την αρχική μεταβλητή. Αυτό πρακτικά λειτουργεί σαν call by reference

**Παράδειγμα 1**

```
#include <stdio.h>

void swap(int *x, int *y) {
   int temp = *x;
   *x = *y;
   *y = temp;
}

int main(void) {
   int a = 5, b = 9;

   printf("Πριν: a = %d, b = %d\n", a, b);
   swap(&a, &b);   // call by "reference" μέσω δεικτών
   printf("Μετά: a = %d, b = %d\n", a, b);

   return 0;
}
```

**Παράδειγμα 2**

```
#include <stdio.h>

void addOne(int *x) {
   *x = *x + 1;   // αλλάζουμε την τιμή της μεταβλητής στην οποία δείχνει ο δείκτης
}

int main(void) {
   int a = 10;
   printf("Πριν: a = %d\n", a);

   addOne(&a);    // περνάμε τη διεύθυνση του a

   printf("Μετά: a = %d\n", a);
   return 0;
}
```

- int *x είναι δείκτης σε int, δηλαδή μια διεύθυνση μνήμης όπου βρίσκεται ένας int.

- &a σημαίνει «η διεύθυνση του a».

- *x σημαίνει «η τιμή στην οποία δείχνει ο x», οπότε αλλάζοντας *x αλλάζεις τον a στον main.

## Δυναμική δέσμευση μνήμης

Η malloc δεσμεύει δυναμικά χώρο στη heap κατά την εκτέλεση και επιστρέφει έναν δείκτη στην αρχή αυτού του χώρου. Πρέπει πάντα να αποδεσμεύεις αυτόν τον χώρο με free όταν δεν τον χρειάζεσαι.

**Βασική χρήση malloc**
Βήματα:

1. Υπολογίζεις πόσα bytes χρειάζεσαι.
2. Καλείς malloc(μέγεθος_σε_bytes).
3. Ελέγχεις αν επέστρεψε NULL (αποτυχία).
4. Χρησιμοποιείς τη μνήμη μέσω του δείκτη.
5. Καλείς free(ptr) όταν τελειώσεις.

```
#include <stdio.h>
#include <stdlib.h>   // για malloc, free

int main(void) {
   int n;
   printf("Δώσε πλήθος στοιχείων: ");
   scanf("%d", &n);

   // δέσμευση μνήμης για n ακέραιους
   int *arr = malloc(n * sizeof(int));
   if (arr == NULL) {
      printf("Αποτυχία δέσμευσης μνήμης\n");
      return 1;
   }

   // γέμισμα του πίνακα
   for (int i = 0; i < n; i++) {
      arr[i] = i * 10;
   }

   // εκτύπωση
   for (int i = 0; i < n; i++) {
      printf("arr[%d] = %d\n", i, arr[i]);
   }

   // αποδέσμευση μνήμης
   free(arr);
   arr = NULL;  // καλό να μη μένει κρεμασμένος δείκτης
   return 0;
}
```