Η γλώσσα προγραμματισμού C

Οι συναρτήσεις στη C

Κ. Βασιλάκης, ΣΤΕΦ, ΤΕΙ Κρήτης

Οι συναρτήσεις – τι είναι

- Πρόκειται για ανεξάρτητα τμήματα ενός προγράμματος (υπο-προγράμματα) που επιτελούν συγκεκριμένες εργασίες.
- Καλούνται από το κυρίως πρόγραμμα ή από άλλες συναρτήσεις (ακόμα και τον εαυτό τους) και μπορούν να παράγουν αποτελέσματα με χρήση, όπου είναι σκόπιμο, δεδομένων.
- Η χρήση συναρτήσεων μας δίνει τη δυνατότητα να επιμερίσουμε ένα πολύπλοκο πρόγραμμα σε μικρότερα αυτόνομα τμήματα κώδικα (modules) που αναλαμβάνουν μεμονωμένες επιλύσεις επί μέρους προβλημάτων.
- Οι συναρτήσεις είναι δομικά στοιχεία της C (τα προγράμματα της, είναι και αποτελούνται από, συναρτήσεις).
- Όλα τα προγράμματα που έχουμε δει έως τώρα περιέχουν τουλάχιστον μία συνάρτηση: την ***main()*** που είναι το κυρίως πρόγραμμα.

Οι συναρτήσεις – που βρίσκονται

- Υπάρχουν έτοιμες (προ-μεταγλωττισμένες ή όχι) που περιλαμβάνονται σε βιβλιοθήκες, οι οποίες ενσωματώνονται με κατάλληλες εντολές στο κυρίως πρόγραμμα (#include).
- Μέσα στο κυρίως πρόγραμμα όπου δηλώνονται και συντάσσονται σαν ξεχωριστές ενότητες (δημιουργούνται από τους προγραμματιστές).
- Επίσης, μπορεί να βρίσκονται σε αρχεία διαφορετικά από αυτό του κυρίως προγράμματος.
- Σε κάθε περίπτωση είναι μέρος του κυρίως προγράμματος και μεταγλωττίζονται μαζί με αυτό (ή ενσωματώνονται στο εκτελέσιμο, αν είναι μεταγλωττισμένες).
- Η κλήση τους γίνεται με το όνομα τους, που είναι μοναδικό.
- Είναι δυνατόν να κληθούν πολλές φορές και με διαφορετικά δεδομένα.

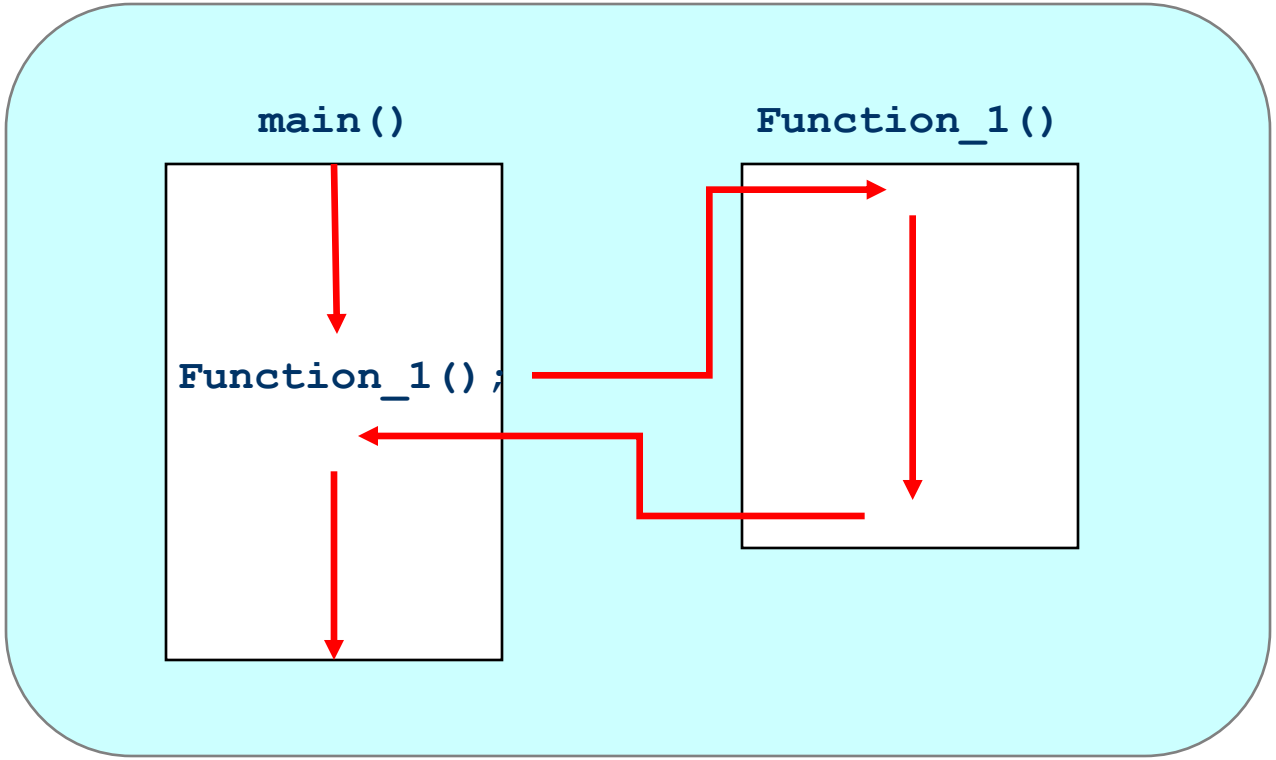
Οι συναρτήσεις – πλεονεκτήματα

- Τα επιμέρους (υπο) προβλήματα διαχειρίζονται πιο εύκολα (μικρότερα προβλήματα --> πιο εύκολα προβλήματα).
- Ευανάγνωστα προγράμματα.
- Αποφεύγουμε την επανάληψη κώδικα.
- Μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις και σε άλλα προγράμματα (επαναχρησιμοποίηση).
- Αφαίρεση (κρύβουν λεπτομέρειες που δεν είναι πάντα απαραίτητες).
- Αν διαγραφούν από τον κώδικα, απενεργοποιείται μόνο η συγκεκριμένη λειτουργία που κάνουν.
- Μπορούν να μεταγλωττιστούν ανεξάρτητα.
- Μεγαλύτερη ευκολία στον έλεγχο και την διόρθωση του κώδικα.
- Μειωμένο κόστος ανάπτυξης.
- Η συντήρηση (μετατροπές/βελτιώσεις) των προγραμμάτων καθίσταται πιο εύκολη.
- Εφαρμογή του ***δομημένου προγραμματισμού***.

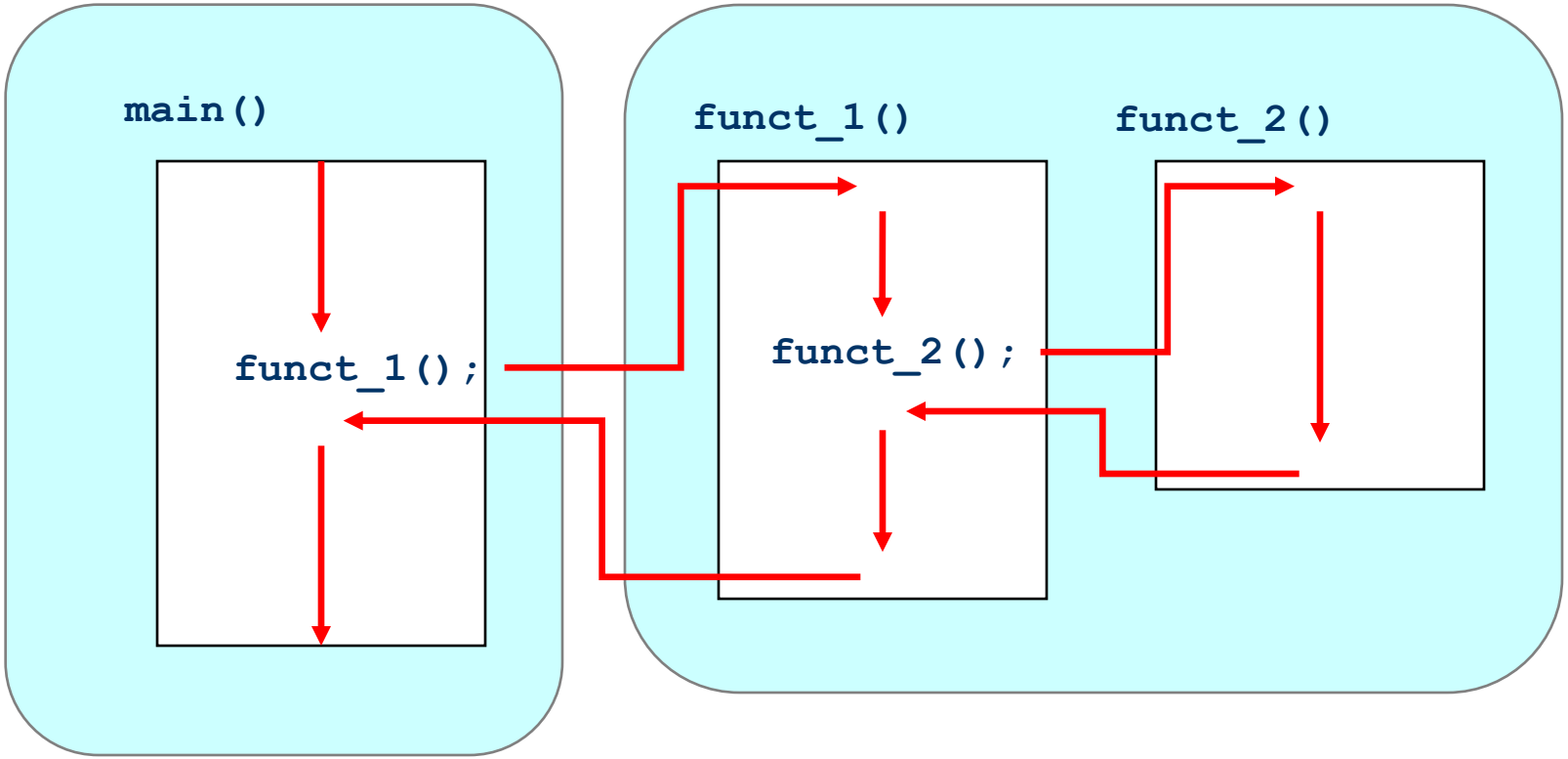
Συναρτήσεις - χρήση

- Κάθε συνάρτηση μπορεί να δεχθεί δεδομένα και να επιστρέψει αποτελέσματα στο πρόγραμμα που την καλεί.
- Για την ανταλλαγή των δεδομένων στοιχείων χρησιμοποιούνται παράμετροι (δηλ. μεταβλητές).
- Υπάρχουν διάφορα είδη συναρτήσεων:
  - Αυτές που δεν χρησιμοποιούν παραμέτρους και δεν επιστρέφουν κάτι.
  - Αυτές που δεν χρησιμοποιούν παραμέτρους και επιστρέφουν κάτι.
  - Αυτές που χρησιμοποιούν παραμέτρους και δεν επιστρέφουν κάτι.
  - Αυτές που χρησιμοποιούν παραμέτρους και επιστρέφουν κάτι.
- Χαρακτηριστικά:
  - Η είσοδος δεδομένων γίνεται από ένα μόνο σημείο.
  - Όταν καλείται μια συνάρτηση σταματά η εκτέλεση του κυρίως προγράμματος (αναμονή) και εκτελείται μόνο η συνάρτηση.
  - Όταν ολοκληρωθεί η εκτέλεση της συνάρτησης, ο έλεγχος επιστρέφει στο πρόγραμμα που έκανε την κλήση της.

Ροή προγράμματος όταν καλείται μια συνάρτηση



Ροή προγράμματος με κλήση συναρτήσεων



Ορισμός μιας συνάρτησης στη C

- Η γενική δομή **ορισμού** μιας συνάρτησης σε C περιλαμβάνει:
  - την επικεφαλίδα και
  - το σώμα της συνάρτησης
- Η γενική δομή ορισμού έχει ως εξής:

τύπος_επιστροφής όνομα_συνάρτησης (τυπικοί παράμετροι)

{

Δηλώσεις μεταβλητών; Τοπικές μεταβλητές

Εντολές συνάρτησης; Εντολές που επιτελούν συγκεκριμένη εργασία

return <εκφραση> ; Τιμή που επιστρέφει η συνάρτηση

}

- Ο ορισμός μιας συνάρτησης ΔΕΝ μπορεί να γίνει μέσα σε άλλη συνάρτηση. Μπορεί να γίνει πριν ή μετά τη ***main()***.

Η επικεφαλίδα

τύπος_επιστροφής όνομα_συνάρτησης (τυπικοί παράμετροι)

- Ο ***τυπος_επιστροφής*** δηλώνει τον τύπο της τιμής (το πολύ μία) που επιστρέφει η συνάρτηση με την εντολή ***return***.
- Αν η συνάρτηση δεν επιστρέφει κάποια τιμή (διαδικασία) τότε ο τύπος επιστροφής είναι τύπου ***void***.
- Το ***όνομα_συνάρτησης*** είναι μοναδικό.
- Οι ***τυπικοί παράμετροι*** (μεταβλητές) είναι ουσιαστικά ο μηχανισμός επικοινωνίας της συνάρτησης με τη συνάρτηση (η ***main*** ή κάποια άλλη) που την καλεί και είναι μια λίστα της μορφής:

  τυπος παραμετρος-1, τύπος παράμετρος-2, … τυπος παραμετρος-n

- Αν δεν υπάρχουν παράμετροι χρησιμοποιείται ο τύπος ***void***.
- Παράδειγμα ορισμού:

τύπος_επιστροφής: ***int*** Όνομα: ***find_max*** τύπος: ***(int*** παράμετρος: ***n, int m)***

Το σώμα της συνάρτησης

- Πρόκειται για κώδικα που περιέχεται σε άγκιστρα και περιλαμβάνει δηλώσεις μεταβλητών και εντολές.
- Εκτελείται όταν κληθεί η συνάρτηση. Μόνο η ***main()*** εκτελείται αυτόματα.
- Οι μεταβλητές που δηλώνονται ισχύουν μόνο μέσα στο σώμα της συνάρτησης. Το ίδιο ισχύει και για τις παραμέτρους.
- Η εκτέλεση τερματίζεται με εντολές ***return***, οι οποίες προκαλούν άμεση έξοδο από τη συνάρτηση.
- Αν δεν υπάρχει εντολή ***return***, η έξοδος προκαλείται όταν εκτελεστεί η τελευταία εντολή.

```
int  find_max (int n,  int m)
{
    int max;
    if (n>m)        max=n;
    else      max=m;
    return max;   // επιστροφή τιμής
}
```

Κλήση μιας συνάρτησης

- Με τη κλήση μιας συνάρτησης, ο έλεγχος του προγράμματος μεταφέρεται στο **σώμα** της συνάρτησης.
- Όταν ολοκληρωθεί η εκτέλεση της συνάρτησης, ο έλεγχος του προγράμματος επιστρέφει στη συνάρτηση που έκανε τη κλήση.
- Μια συνάρτηση καλείται με το όνομα της και τις τιμές που δίνουμε στις μεταβλητές των τυπικών παραμέτρων.
- Αυτές οι τιμές ονομάζονται ***ορίσματα*** (πραγματικοί παράμετροι) και πρέπει να είναι του ίδιου τύπου με τις μεταβλητές των τυπικών παραμέτρων.
- Όταν καλείται μια συνάρτηση πρέπει να είναι γνωστός ο τύπος επιστροφής:

max: ακέραιος

max = find_max (a, b);

Τύπος επιστροφής: int

max = find_max (5, 6);

Παράδειγμα ορισμού και κλήσης συνάρτησης

```
#include <stdio.h>
int find_max (int n,  int m) // Ορισμός της συνάρτησης find_max
{
    int max;  //τοπική  μεταβλητή
    if(n>m)  max=n;
    else       max=m;
    return max;   // επιστροφή τιμής
}
main()   // κυρίως πρόγραμμα
{
  int a,b, max;
  printf ("a:"); scanf("%d",&a);
  printf ("b:"); scanf("%d",&b);
  max= find_max (a,b);  // κλήση της συνάρτησης find_max
  printf("Max of %d and %d is %d\n", a, b, max);
}
```

εμβέλεια μέσα στη συνάρτηση

Μεταβλητές της main.
max διαφορετική από αυτή της συνάρτησης

Δήλωση μιας συνάρτησης (πρωτότυπο/αρχέτυπο)

- Απαιτείται αν πρόκειται να χρησιμοποιήσουμε μια συνάρτηση πριν τον ορισμό της.
- Αυτό συμβαίνει όταν ο ορισμός γίνεται μετά τη συνάρτηση ***main()*** ή βρίσκεται σε διαφορετικό αρχείο ή η συνάρτηση είναι ήδη μεταγλωττισμένη.
- Η δήλωση του πρωτοτύπου μιας συνάρτησης προσδιορίζει τον τύπο επιστροφής της συνάρτησης, το όνομα της και τους τύπους των παραμέτρων της:

```
int  find_max (int, int) ;
```

Τελειώνει με το ;

Δεν χρειάζονται ονόματα παραμέτρων

δεν υπάρχει σώμα.

- Μια καλή πρακτική είναι οι δηλώσεις των συναρτήσεων να γίνονται πριν την ***main()***. Αν βρίσκονται σε άλλο αρχείο θα πρέπει να συμπεριλαμβάνονται στο πρόγραμμα με την οδηγία ***#include***.

Παράδειγμα δήλωσης, κλήσης και ορισμού συνάρτησης

```
#include <stdio.h>
int find_max(int, int); //Δήλωση της συνάρτησης - πρωτότυπο
main()
{
  int a,b, max;
  printf ("a:"); scanf("%d",&a);
  printf ("b:"); scanf("%d",&b);
  max= find_max(a,b); // Κλήση της συνάρτησης
  printf("Max of %d and %d is %d\n", a, b, max);
}
int find_max (int n, int m) //Ορισμός συνάρτησης
{
   int max;
   if(n>m)   max=n;
   else   max=m;
   return max;
}
```

```
printf("Max of %d and %d is %d\n", a, b, find_max(a,b));
```

Αλλά και:

```
int find_max (int n, int m) {
      if(n>m) return n;
      else    return m;
}
```

Συναρτήσεις ΧΩΡΙΣ παραμέτρους και ΧΩΡΙΣ επιστροφή

ΔΗΛΩΣΗ

ΚΛΗΣΗ

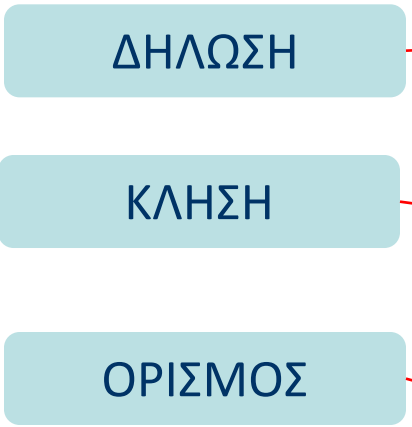
ΟΡΙΣΜΟΣ

```
#include <stdio.h>
.....
void printstarline(void);
/* void printstarline(); */
.............
main() {
.............
for (i=1; i<=N;i++)
                printstarline();

.......... }
void printstarline(void)
{
    int i;
   for (i=1; i<=80;i++)
            printf ("%c", '*');
}
```

- Δεν μεταβιβάζεται τιμή στη συνάρτηση.
- Η συνάρτηση απλά κάνει κάποια εργασία. Δεν επιστρέφει τιμή.

Συναρτήσεις ΧΩΡΙΣ παραμέτρους ΜΕ επιστροφή




```
float findsum(void);

main() {
        ……….
        athroisma = findsum();
        MO = athroisma/7;
        …….
}
float findsum(void) {
    int i;
    float a, sum=0;
    printf ("You should give 7 numbers\n");
    for (i=1; i<=7;i++) {
        printf("Please give %d number:",i);
        scanf("%f", &a);
        sum=sum+a;
     }
     return sum;
}
```

- Δεν μεταβιβάζεται τιμή στη συνάρτηση.
- Η συνάρτηση επιστρέφει κάποια τιμή.

Συναρτήσεις ΜΕ παραμέτρους ΧΩΡΙΣ επιστροφή

```
#include <stdio.h>
……..
void printchar (int, char);
……..
main() {
……..
C='*';
for (i=1; i<=N;i++)
        printchar (i, C);
……..}
void printchar (int m, char C )
{
  int i;
  for (i=1; i<=m;i++)
        printf ("%c", C);
  printf ("\n");
}
```

ΔΗΛΩΣΗ

ΚΛΗΣΗ

ΟΡΙΣΜΟΣ

- Μεταβιβάζονται τιμές (ορίσματα) στη συνάρτηση.
- Η συνάρτηση δεν επιστρέφει κάποια τιμή.

Συναρτήσεις ΜΕ παραμέτρους και ΜΕ επιστροφή

```
#include <stdio.h>
#define MAXN 12
int compute_factorial (int);
main()
{
  int i;
  for (i=1 ; i <= MAXN ; i++)
    printf("%2d! = %d\n", i, compute_factorial(i));
  getchar();getchar();
}
int compute_factorial (int n)
{
  int i, factorial=1;
  for (i=1; i<=n; i++)
     factorial *= i;
  return factorial;
}
```

ΔΗΛΩΣΗ

ΚΛΗΣΗ

ΟΡΙΣΜΟΣ

- Μεταβιβάζονται τιμές (ορίσματα) στη συνάρτηση.
- Η συνάρτηση επιστρέφει κάποια τιμή.

Μεταβίβαση δεδομένων (κατ' αξία και κατ' αναφορά)

- **Παράμετροι**: οι μεταβλητές που εμφανίζονται στη δήλωση και στον ορισμό της συνάρτησης.
- **Ορίσματα**: οι τιμές που περιέχονται στην κλήση της συνάρτησης.
- Μεταβίβαση δεδομένων **κατ' αξία** (by value), όπου:
  - αντιγράφονται οι τιμές των ορισμάτων στις παραμέτρους που είναι τοπικές μεταβλητές της συνάρτησης και τυχόν αλλαγές δεν επηρεάζουν στις μεταβλητές της συνάρτησης που κάνει τη κλήση.
- Μεταβίβαση δεδομένων **κατ' αναφορά** (by reference), όπου:
  - δεν μεταβιβάζονται οι αξίες, αλλά οι διευθύνσεις των μεταβλητών που περιέχουν τις τιμές, έτσι η συνάρτηση έχει πρόσβαση στις διευθύνσεις των μεταβλητών και μπορεί να κάνει αλλαγές στις τιμές.
- Καλώντας κάποια συνάρτηση κατ' αναφορά έχουμε τη δυνατότητα ν' αλλάξουμε πολλές τιμές μεταβλητών (άρα και να έχουμε επιστροφή πολλών τιμών).

Παράδειγμα κλήσεων κατ' αξία και αναφορά

```
#include <stdio.h>
void swap (int, int);
main()
{
  int x = 1, y = 2;
  printf("Before: x=%d, y=%d.\n", x, y);
  swap (x, y);
  printf("After: x=%d, y=%d. \n", x, y);
  getchar();getchar();
}
void  swap (int u, int v)
{
    int temp;
    temp = u;
    u = v;
    v = temp;
}
```

ΔΕΝ ΑΛΛΑΖΟΥΝ ΟΙ ΤΙΜΕΣ ΤΩΝ ΜΕΤΑΒΛΗΤΩΝ

```
#include <stdio.h>
void swap (int*, int*);
main()
{
  int x = 1, y = 2;
  printf("Before: x=%d, y=%d.\n", x, y);
  swap(&x, &y);
  printf("After: x=%d, y=%d. \n", x, y);
  getchar();getchar();
}
void swap(int *u, int *v)
{
  int temp;
  temp = *u;
  *u = *v;
  *v = temp;
}
```

ΑΛΛΑΖΟΥΝ ΟΙ ΤΙΜΕΣ ΤΩΝ ΜΕΤΑΒΛΗΤΩΝ

Εμβέλεια μεταβλητών

- Εμβέλεια μιας μεταβλητής είναι το τμήμα του προγράμματος που η μεταβλητή ισχύει (μπορεί ν' αξιοποιηθεί).
- Αναφορικά με την εμβέλεια τους οι μεταβλητές κατηγοριοποιούνται σε:
  - ***Τοπικές*** (local) μεταβλητές και
  - ***Καθολικές*** (global) μεταβλητές που δηλώνονται εκτός συναρτήσεων και έχουν εμβέλεια σε όλο το πρόγραμμα.
- Η κατηγοριοποίηση αυτή εξαρτάται από το σημείο που δηλώνεται κάποια μεταβλητή:
  - Οι τοπικές μεταβλητές δηλώνονται **μέσα σε μία συνάρτηση** (στο σώμα της) και έχουν εμβέλεια μόνο μέσα σε αυτή.
  - Οι καθολικές μεταβλητές δηλώνονται **εκτός συναρτήσεων** (συνήθως στη αρχή του αρχείου του κώδικα) και έχουν εμβέλεια σε όλο το πρόγραμμα.

Παρατηρήσεις στη εμβέλεια των μεταβλητών - Ι

- Τοπικές μεταβλητές μπορούν να δηλωθούν και σε κάποιο τμήμα κώδικα (block που αρχίζει και τελειώνει με άγκιστρα: { …. }).
- Η τοπική μεταβλητή έχει εμβέλεια μόνο μέσα στο συγκεκριμένο τμήμα που έχει δηλωθεί.
- Η απόδοση αρχικών τιμών σε τοπικές μεταβλητές είναι μέριμνα του προγραμματιστή (δεν αποδίδονται αυτόματα αρχικές τιμές).
- Σε μια συνάρτηση εκτός των μεταβλητών που δηλώνονται μέσα στο σώμα της συνάρτησης, τοπικές μεταβλητές είναι και οι παράμετροι στον ορισμό της συνάρτησης.
- Οι παράμετροι δεν χρειάζονται αρχική τιμή διότι παίρνουν τιμές όταν καλείται η συνάρτηση (από τις τιμές των ορισμάτων).
- Οι τοπικές μεταβλητές μπορούν να έχουν το ίδιο όνομα σε διαφορετικές συναρτήσεις. Είναι όμως διαφορετικές μεταβλητές.

Παρατηρήσεις στη εμβέλεια των μεταβλητών - ΙΙ

- Οι καθολικές μεταβλητές έχουν εμβέλεια σε όλο το πρόγραμμα και κάθε συνάρτηση του προγράμματος μπορεί να τις χρησιμοποιήσει.
- Στις καθολικές μεταβλητές αποδίδεται αυτόματα η τιμή 0 (εκτός αν ο προγραμματιστής αναθέσει κάποια άλλη αρχική τιμή).
- Οι τοπικές και οι καθολικές μεταβλητές μπορούν να έχουν το ίδιο όνομα, αλλά είναι διαφορετικές.
- Σε περίπτωση που έχουμε μια τοπική και μια καθολική μεταβλητή με το ίδιο όνομα, στο τμήμα (εμβέλεια) που έχει δηλωθεί η τοπική μεταβλητή, αυτή υπερισχύει της καθολικής.
- Η εμβέλεια μια καθολικής μεταβλητής ξεκινά από το σημείο που δηλώνεται αυτή και όλες οι συναρτήσεις που ορίζονται μετά από τη δήλωση της μπορούν να τη χρησιμοποιήσουν.

Παράδειγμα με τοπικές και καθολικές μεταβλητές

```
#include <stdio.h>
int find_max(int, int);
main()
{
  int a,b, max;
  printf ("a:"); scanf("%d",&a);
  printf ("b:"); scanf("%d",&b);
  max= find_max(a,b);
  printf("Max is: %d\n", a, b, max);
 getchar();getchar();
}
int find_max (int n, int m) {
      int max;
      if(n>m)   max=n;
      else   max=m;
      return max;
}
```

Όλες οι μεταβλητές είναι τοπικές

2 διαφορετικές max

```
#include <stdio.h>
int max;
int find_max(int, int);
main()
{
  int a,b;
  printf ("a:");scanf("%d",&a);
  printf ("b:");scanf("%d",&b);
  find_max(a,b);
  printf("Max is:%d\n",max);
  getchar();getchar();
}
int find_max(int n, int m)  {
      if(n>m)  max=n;
      else  max=m;
}
```

max: καθολική

a, b : τοπικές

n, m : τοπικές

Δίνουμε τιμές στη καθολική max

Χωρίς return!

Μεταβλητές static (τοπικές)

- Το χαρακτηριστικό γνώρισμα μιας static τοπικής μεταβλητής είναι ότι μπορεί να διατηρήσει τη τιμή της σε διαδοχικές κλήσεις της συνάρτησης.

static int i;

- Αυτό είναι χρήσιμο όταν θέλουμε να διατηρήσουμε τη τιμή μιας τοπικής μεταβλητής ανάμεσα σε κλήσεις της συνάρτησης.
- Η static τοπική μεταβλητή παίρνει τιμή όταν καλείται για πρώτη φορά η συνάρτηση και διατηρεί τη τελευταία της τιμή στις επόμενες κλήσεις της συνάρτησης.
- Κρατά τη ίδια θέση μνήμης κατά τη διάρκεια της εκτέλεσης του προγράμματος.
- Ισχύει όμως μόνο μέσα στη εμβέλειας της (στο τμήμα που έχει δηλωθεί).

Παράδειγμα με static μεταβλητή

```
#include <stdio.h>
#define N 5
void func(void);
main() {
   int i;
   for (i=1; i<=N; i++)
                func();
          getchar();getchar();
}
void func()
{
          static int x = 1;
          int i;
          for (i=1;i<=x;i++)
                    printf("%c", '*');
          x = x + 1;
          printf("\n");
}
```

Δήλωση

Κλήση

Ορισμός

x: static

Η τιμή της x διατηρείται και στη επόμενη κλήση

Αποτέλεσμα εκτέλεσης

```
*
**
***
****
*****
```

Μεταβλητές extern (καθολικές)

- Χρησιμοποιούνται όταν η εμβέλεια μιας καθολικής μεταβλητής πρέπει να περιλαμβάνει πολλά αρχεία ενός προγράμματος.
- Ουσιαστικά δηλώνεται ξανά μια καθολική μεταβλητή, η οποία έχει οριστεί σε άλλο αρχείο του προγράμματος.
- Βρίσκει εφαρμογή σε μεγάλα προγράμματα που εκτείνονται σε πολλά αρχεία.
- Η εντολή:

  extern int global;

  ενημερώνει το μεταγλωττιστή ότι η μεταβλητή global έχει οριστεί σε άλλο αρχείο του προγράμματος.
- Όλα τα αρχεία του προγράμματος στα οποία δηλώνεται η εξωτερική (extern) μεταβλητή μπορούν να την επεξεργαστούν.

Καποιες συναρτήσεις της βιβλιοθηκης <math.h>

- double sqrt(double x); //τετραγωνική ρίζα του x (πρέπει x > 0)
- double exp(double x); //εκθετικό του x, e^x
- double log(double x); // φυσικός λογάριθμος του x, ln(x)
- int abs(int x); // απόλυτη τιμή του val
- double pow(double x, double y); // το x^y
- double sin(double x); // ημίτονο του x (σε rad)
- double cos(double x); // συνημίτονο του x (σε rad)
- double tan(double x); //εφαπτομένη του x (σε rad)
- double ceil(double x); //η μικρότερη ακέραια τιμή όχι μικρότερη από x.
- double floor(double x); // η μεγαλύτερη ακέραια τιμή όχι μεγαλύτερη από x.