

Κεφάλαιο

9

Συναρτήσεις



Συναρτήσεις

Σε αυτό το κεφάλαιο θα αναφερθούμε στη χρήση των συναρτήσεων από τη C. Μέχρι στιγμής είδαμε τη χρήση της `main()`, μιας συνάρτησης απαραίτητης σε κάθε πρόγραμμα της C. Επίσης συναντήσαμε και ορισμένες συναρτήσεις βιβλιοθήκης όπως οι `printf()`, `getch()` κ.ά.

Οι συναρτήσεις βοηθούν τον προγραμματιστή να "κομματιάσει" ένα μεγάλο πρόγραμμα σε μικρότερα τμήματα, κάθε ένα από τα οποία επιτελεί μια συγκεκριμένη λειτουργία.

Στις δομημένες γλώσσες προγραμματισμού, οι συναρτήσεις μπορούν να "κρύβουν" κομμάτια κώδικα από το υπόλοιπο πρόγραμμα με τρόπο ώστε οι μεταβλητές μιας συνάρτησης να μην επηρεάζουν το υπόλοιπο πρόγραμμα.

Όπως αναφέραμε στην εισαγωγή, ένα πρόγραμμα στη C έχει την ακόλουθη γενική μορφή:

```
main()
{
    δηλώσεις μεταβλητών της συνάρτησης main();
    εκτελέσιμες προτάσεις της συνάρτησης main();
}

function-1()
{
    δηλώσεις μεταβλητών της συνάρτησης function-1();
    εκτελέσιμες προτάσεις της συνάρτησης function-1();
}

function-2()
{
    δηλώσεις μεταβλητών της συνάρτησης function-2();
    εκτελέσιμες προτάσεις της συνάρτησης function-2();
}

function-N()
{
    δηλώσεις μεταβλητών της συνάρτησης function-N();
```

```

    εκτελέσιμες προτάσεις της συνάρτησης function-N();
}

```

Οι συναρτήσεις εκτελούνται μόνο αν κληθούν από μια άλλη συνάρτηση. Η μόνη συνάρτηση που εκτελείται πάντοτε είναι η συνάρτηση **main()**.

Ορισμός μιας συνάρτησης

Μια συνάρτηση ορίζεται με τον επόμενο τρόπο:

τύπος_συνάρτησης όνομα_συνάρτησης (παράμετρος1,παράμετρος2,...)


δηλώσεις παραμέτρων συνάρτησης;

```
{
```

δηλώσεις μεταβλητών συνάρτησης;

εκτελέσιμες προτάσεις συνάρτησης;

```
}
```

 Ο τύπος της συνάρτησης καθορίζει τον τύπο της τιμής που επιστρέφει η συνάρτηση.

 Αν δεν οριστεί ο τύπος της συνάρτησης, η C υποθέτει ότι επιστρέφει τιμή τύπου **int**.

Στη συνέχεια ορίζεται μια συνάρτηση με όνομα **mo()**, η οποία υπολογίζει και επιστρέφει το μέσο όρο δύο ακέραιων αριθμών:

```


float mo(x, y)
int x, y;
{
    float m;
    m = (x+y)/2.0;
    return(m);
}

```

Η συνάρτηση είναι τύπου **float**, που σημαίνει ότι θα επιστρέψει ως τιμή έναν αριθμό τύπου **float** (κινητής υποδιαστολής). Έχει δύο παραμέτρους (**x** και **y**) μέσω των οποίων το πρόγραμμα που καλεί τη συνάρτηση, της διαβιβάζει πλη-

ροφορίες. Οι παράμετροι στο συγκεκριμένο παράδειγμα είναι τύπου `int` και δηλώνονται αμέσως μετά τη δήλωση της συνάρτησης και πριν από την αριστερή της αγκύλη `{`.


Με την πρόταση `return (m)` η συνάρτηση επιστρέφει τον έλεγχο στο πρόγραμμα που την κάλεσε και μάλιστα επιστρέφει ως τιμή το περιεχόμενο της `m`.

 Στην περίπτωση που δεν υπάρχει στο σώμα της συνάρτησης εντολή `return`, η συνάρτηση επιστρέφει μόλις εκτελέσει και την τελευταία της πρόταση χωρίς όμως να επιστρέψει συγκεκριμένη τιμή.

Ένας εναλλακτικός τρόπος δήλωσης μιας συνάρτησης είναι ο εξής:

τύπος-συνάρτησης όνομα-συνάρτησης (τύπος παράμετρος1, τύπος παράμετρος2, ...)

```
{
    δηλώσεις μεταβλητών συνάρτησης;
    εκτελέσιμες προτάσεις συνάρτησης;
}
```


 Με τον παραπάνω τρόπο ο τύπος των μεταβλητών αναφέρεται μέσα στις παρενθέσεις μαζί με τα ονόματα των παραμέτρων.

Επομένως, η συνάρτηση `mo ()` μπορεί να δηλωθεί και ως εξής:

```
float mo(int x, int y)
{
    float m;
    m = (x+y)/2.0;
    return (m);
}
```

Συναρτήσεις που επιστρέφουν τιμή

Όπως αναφέραμε προηγουμένως, μια συνάρτηση μπορεί να επιστρέφει μία τιμή.

 Αν στον ορισμό της συνάρτησης δεν έχει οριστεί ο τύπος της, τότε η C υποθέτει ότι επιστρέφει μία τιμή τύπου `int`.

👉 Όταν η συνάρτηση επιστρέφει τιμή άλλου τύπου (π.χ. **float**) τότε **πρέπει οπωσδήποτε** να δηλωθεί ο τύπος της.

Η συνάρτηση επιστρέφει μία τιμή στο πρόγραμμα που την κάλεσε μέσω της εντολής **return**. Η σύνταξη της **return** είναι η εξής:

return(παράσταση); ή

return παράσταση;

Η **return** τερματίζει τη συνάρτηση και επιστρέφει ως τιμή το αποτέλεσμα της παράστασης.

👉 Το πρόγραμμα που καλεί μία συνάρτηση μπορεί να αγνοήσει την τιμή που επιστρέφεται από τη συνάρτηση.

Το επόμενο πρόγραμμα καλεί τη συνάρτηση **mo()** δύο φορές. Την πρώτη φορά δεν χρησιμοποιεί την τιμή που επιστρέφεται και τη δεύτερη την καταχωρίζει στη μεταβλητή **c**.

```
main()
```

```
{
```

```
    int a=10,b=3;
```

```
    float c;
```

```
    mo(a,b);
```

```
    c=mo(a,b);
```

```
}
```

Ο αριθμός που επιστρέφει η **mo()** χάνεται

Ο αριθμός που επιστρέφει η **mo()** καταχωρίζεται στη μεταβλητή **c**.

Συναρτήσεις που δεν επιστρέφουν τιμή

Μια συνάρτηση μπορεί να μην επιστρέφει συγκεκριμένη τιμή αλλά απλώς να εκτελεί μια συγκεκριμένη λειτουργία.

```
display(x,y)
```

```
int x,y;
```

```
{
```


```
    int sum;
```

```
    sum = x+y;
```

```
    printf("το άθροισμα %d και %d είναι %d\n",x,y,sum);
```


```
}
```

Η παραπάνω συνάρτηση δεν επιστρέφει καμία τιμή και επιστρέφει στο πρόγραμμα που την κάλεσε αφού εκτελέσει και την τελευταία της πρόταση.

 Η πρόταση **return** χωρίς παράσταση (και χωρίς παρενθέσεις) χρησιμοποιείται για έξοδο από μια συνάρτηση πριν φτάσει στο φυσικό της τέλος και χωρίς να επιστρέψει τιμή.


Για παράδειγμα, η συνάρτηση που ακολουθεί επιστρέφει στο πρόγραμμα που την καλεί (χωρίς συγκεκριμένη τιμή), όταν οι παράμετροι έχουν την ίδια τιμή.

```
display(x,y)
int x,y;
{
    int sum;
    if (x==y) return;
    sum = x+y;
    printf("το άθροισμα %d και %d είναι %d\n",x,y,sum);
}
```

 Παρόλο που οι παραπάνω συναρτήσεις δεν επιστρέφουν τιμή, ο μεταγλωττιστής της C θεωρεί ότι επιστρέφουν μια τιμή τύπου **int** (επειδή δεν έχει δηλωθεί κανένας τύπος στον ορισμό τους). Έτσι επιτρέπει προτάσεις της μορφής:

```
a = display(4,5) + 8;
```

που δεν έχουν βέβαια νόημα, αφού η **display()** δεν επιστρέφει καμία τιμή.

 Αν θέλουμε ο μεταγλωττιστής (compiler) της C να γνωρίζει ότι μια συνάρτηση δεν επιστρέφει τιμή πρέπει να την ορίσουμε ως τύπου **void**.

```
void display(x,y)
int x,y;
{
    int sum;
    sum = x+y;
    printf("το άθροισμα %d και %d είναι %d\n",x,y,sum);
}
```


Αν η συνάρτηση έχει δηλωθεί ως τύπου **void**, ο μεταγλωττιστής γνωρίζει ότι δεν επιστρέφει καμία τιμή και θεωρεί λανθασμένες προτάσεις της μορφής:

```
a = display(4,5) + 8;
```

 Η δήλωση μιας συνάρτησης ως τύπου **void**, δεν αλλάζει τίποτα στη λειτουργία του προγράμματος αλλά μας προστατεύει από λανθασμένη χρήση της.

Η κλήση μιας συνάρτησης

Όταν ένα πρόγραμμα καλεί μια συνάρτηση, υποθέτει ότι η συνάρτηση είναι τύπου **int**.

 Αν η συνάρτηση που καλείται δεν είναι τύπου **int**, πρέπει να δηλωθεί ξανά μέσα στο πρόγραμμα πριν κληθεί. Τη δήλωση αυτή την ονομάζουμε *πρόσθια δήλωση* (forward declaration).

```
main()
{
    int a,b;
    float mo();
    a=23;
    b=2;
    printf("%f\n",mo(a,b));
}
```

Στο σημείο αυτό, η **mo()** δηλώνεται ως τύπου **float** για να μπορεί να χρησιμοποιηθεί σωστά από τη **main()**.

```
float mo(x,y)
int x,y;
{
    float m;
    m = (x+y)/2.0;
    return(m);
}
```

Στο σημείο που δηλώνεται η συνάρτηση, αναφέρεται ο τύπος της και το όνομά της (χωρίς τις παραμέτρους, αλλά με τις παρενθέσεις).

Συνήθως η συνάρτηση δηλώνεται πριν από τη **main()**:

```
float mo();
main()
{
```

```
int a,b;
a=23;
b=2;
printf("%f\n",mo(a,b));
}

float mo(x,y)
int x,y;
{
    float m;
    m = (x+y)/2.0;
    return(m);
}
```

Αυτός ο τρόπος πρόσθιας δήλωσης (forward declaration) γίνεται για να ενημερωθεί ο μεταγλωττιστής ότι αργότερα θα οριστεί μια συνάρτηση **τάδε** τύπου με **δείνα** όνομα.

Μια συνάρτηση μπορεί να οριστεί και πριν από τη **main()**. Αν χρησιμοποιηθεί από τη **main()** ή από άλλη συνάρτηση που ορίζεται μετά, **τότε δε χρειάζεται να ξαναδηλωθεί**. Για παράδειγμα:

```
float mo(x,y)
int x,y;
{
    float m;
    m=(x+y)/2.0;
    return(m);
}

main()
{
    int a,b;
    a=23;
    b=2;
    printf("%f\n",mo(a,b));
}
```


Παράμετροι συνάρτησης

Όπως αναφέραμε προηγουμένως, αν μια συνάρτηση έχει παραμέτρους, αυτές δηλώνονται αμέσως μετά από το όνομα της συνάρτησης (και πριν από το "σώμα" της).

```
int add(x, y)
int x, y;
{
    .....
    .....
}
```


Όνομα συνάρτησης

Δήλωση παραμέτρων

Σώμα συνάρτησης

Υπάρχει και δεύτερος τρόπος δήλωσης των παραμέτρων μιας συνάρτησης, μαζί με το όνομα της συνάρτησης και όχι σε ξεχωριστή πρόταση:

```
int add(int x, int y)
{
    .....
    .....
}
```

 Οι δύο τρόποι δήλωσης των παραμέτρων μιας συνάρτησης είναι ισοδύναμοι

Ορίσματα και διαβίβαση παραμέτρων

Όπως αναφέραμε προηγουμένως, οι παράμετροι μιας συνάρτησης χρησιμοποιούνται για τη διαβίβαση δεδομένων από το πρόγραμμα προς τη συνάρτηση την οποία καλεί.

Οι τιμές με τις οποίες καλείται μια συνάρτηση λέγονται **ορίσματα** της συνάρτησης

Ας θεωρήσουμε το επόμενο παράδειγμα κλήσης της συνάρτησης `add()` από τη συνάρτηση `main()`.

```
main()
{
    int a,b,c;
    a=10;
    b=20;
    c=add(a,b);
    printf("Το άθροισμα είναι %d\n",c);
}

int add(x,y)
int x,y;
{
    int ss;
    ss=x+y;
    return(ss);
}
```

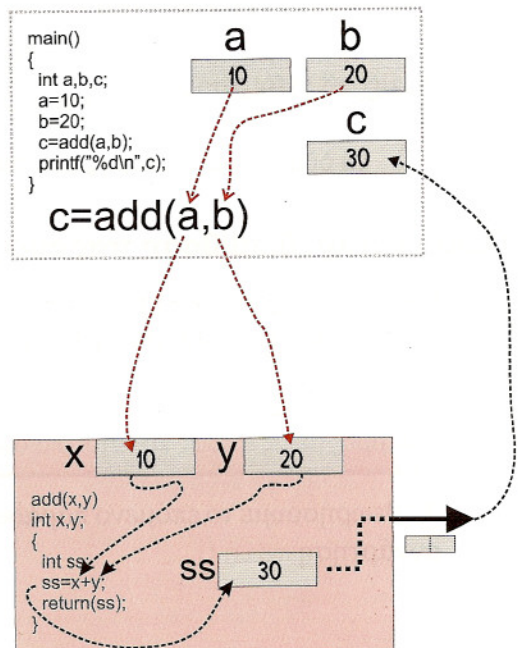
Ορίσματα συνάρτησης

Παράμετροι συνάρτησης

Η συνάρτηση `add()` έχει οριστεί με δύο παραμέτρους `x` και `y`. Όταν καλείται η συνάρτηση, οι τιμές των ορισμάτων της (το 10 και το 20) αντιγράφονται στις παραμέτρους `x` και `y` αντίστοιχα.

Αυτός ο τρόπος διαβίβασης των ορισμάτων λέγεται **κλήση ή μεταβίβαση κατ' αξία** (*call by value*) και είναι ο μόνος τρόπος που υποστηρίζει η C.

Οι μεταβλητές των παραμέτρων που ορίζονται σε μια συνάρτηση (στη περίπτωση μας οι `x` και `y`) λέγονται **τυπικές παράμετροι** (*formal parameters*) της συνάρτησης.





Όταν καλείται μια συνάρτηση, τα ορίσματα πρέπει να είναι ίδιου τύπου με τις τυπικές παραμέτρους της, όπως αυτές έχουν δηλωθεί στον ορισμό της συνάρτησης.

Έτσι, αν η συνάρτηση `add()` κληθεί με ορίσματα που δεν αποδίδουν ακέραιες τιμές, τότε το αποτέλεσμα θα είναι κάποιος απροσδιόριστος αριθμός.

Χρήση συναρτήσεων βιβλιοθήκης

Όπως αναφέραμε και στην εισαγωγή, η C είναι μια λιτή γλώσσα, δεδομένου ότι διαθέτει ελάχιστες εντολές. Η αδυναμία αυτή της C αντισταθμίζεται από το πλήθος συναρτήσεων βιβλιοθήκης που διαθέτει. Έτσι, πολλές λειτουργίες που υλοποιούνται σε άλλες γλώσσες προγραμματισμού με εντολές, υλοποιούνται στη C με χρήση συναρτήσεων βιβλιοθήκης.

Το πρότυπο ANSI καθορίζει πέρα από τον ορισμό της γλώσσας και τα περιεχόμενα της τυπικής βιβλιοθήκης της C. Όμως όλες σχεδόν οι εκδόσεις της C συνοδεύονται από πολλές επιπλέον συναρτήσεις, που δεν έχουν οριστεί μέσα στο πρότυπο ANSI.

Οι συναρτήσεις βιβλιοθήκης ορίζονται κατά ομάδες στα **αρχεία κεφαλίδας** (header files). Τα αρχεία κεφαλίδας είναι αρχεία με προέκταση `.h` τα οποία συνοδεύουν το μεταγλωττιστή της C και στα οποία ορίζονται διάφορες συναρτήσεις. Έτσι, π.χ., στο `math.h` ορίζονται μαθηματικές συναρτήσεις, στο `stdio.h` συναρτήσεις που σχετίζονται με είσοδο-έξοδο σε αρχεία κ.λπ.

Για να μπορέσει να χρησιμοποιηθεί μια συνάρτηση βιβλιοθήκης σε ένα πρόγραμμα, θα πρέπει να συμπεριληφθεί και το αρχείο κεφαλίδας στο οποίο ορίζεται η συνάρτηση. Αυτό γίνεται στην αρχή του προγράμματος, με χρήση της οδηγίας `#include`.

Η "οδηγία μεταγλωττιστή" (compiler directive) `#include` συντάσσεται με τον επόμενο τρόπο:

`#include <αρχείο>`

και αναγκάζει το μεταγλωττιστή να συμπεριλάβει κατά τη φάση της μεταγλώττισης το αρχείο το οποίο ορίζει. Συνήθως ή `#include` χρησιμοποιείται με αρχεία κεφαλίδας.


```
#include <math.h>

main()
{
    double p;
    p=pow(2,3);
    printf("p=%f\n",p);
}
```

Το προηγούμενο πρόγραμμα συμπεριλαμβάνει το αρχείο **math.h**, στο οποίο ορίζεται η συνάρτηση **pow()**.

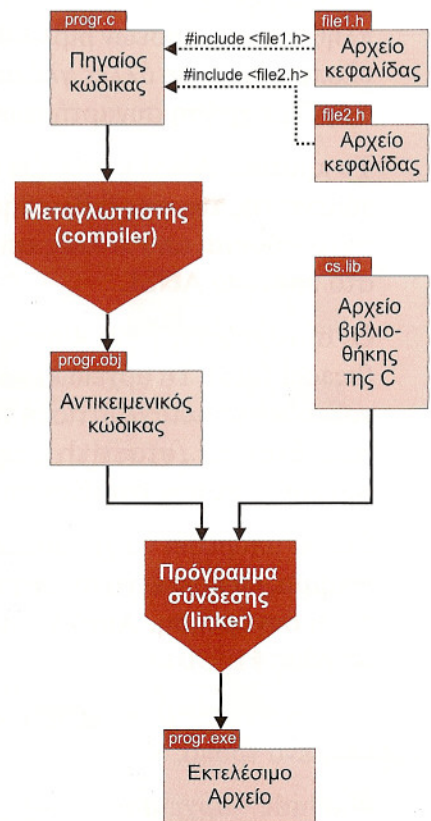
Η οδηγία **#include** δεν είναι εντολή και δεν τερματίζεται με ελληνικό ερωτηματικό (;).

Το διπλανό σχήμα δείχνει όλη τη διαδικασία μεταγλώττισης (μετάφρασης) ενός προγράμματος της C.

Ο μεταγλωττιστής (compiler) συμπεριλαμβάνει κατά τη διαδικασία της μεταγλώττισης στον πηγαίο κώδικα όλα τα αρχεία συμπερίληψης (**#include**) και παράγει το αρχείο που περιέχει τον αντικειμενικό κώδικα (object code).

Ο αντικειμενικός κώδικας είναι το μεταγλωττισμένο σε γλώσσα μηχανής πηγαίο πρόγραμμα, χωρίς όμως να συμπεριλαμβάνει και τον κώδικα των συναρτήσεων βιβλιοθήκης που έχουν χρησιμοποιηθεί.

Το πρόγραμμα σύνδεσης (linker) προσθέτει στον αντικειμενικό κώδικα τον κώδικα των συναρτήσεων βιβλιοθήκης που χρησιμοποιήθηκαν. Ο κώδικας των συναρτήσεων βιβλιοθήκης βρίσκεται σε ένα ή και περισσότερα αρχεία βιβλιοθήκης της C. Το λειτουργικό αποτέλεσμα του προγράμματος σύνδεσης είναι ένα αυτόνομο εκτελέσιμο αρχείο (.exe).



Σύνταξη των συναρτήσεων βιβλιοθήκης

Για να γίνει σωστά η χρήση μιας συνάρτησης βιβλιοθήκης πρέπει να γνωρίζουμε τον τύπο των παραμέτρων της, καθώς και τον τύπο της τιμής που επιστρέφει (αν επιστρέφει κάποια τιμή).

Η σύνταξη μιας συνάρτησης βιβλιοθήκης δίνει όλες τις πληροφορίες για τη σωστή χρήση της. Έτσι, η σύνταξη της συνάρτησης `sqrt()` η οποία υπολογίζει την τετραγωνική ρίζα ενός αριθμού έχει π.χ. ως εξής:

```
#include <math.h>
```

```
double sqrt(num)
```

```
double num;
```

Από τα προηγούμενα συμπεραίνουμε τρία πράγματα:

- Η συνάρτηση ορίζεται στο αρχείο κεφαλίδας **math.h**.
- Η συνάρτηση επιστρέφει τιμή **double**.
- Η συνάρτηση έχει μία παράμετρο τύπου **double**.

Ένας εναλλακτικός τρόπος σύνταξης είναι ο εξής:

```
#include <math.h>
```

```
double sqrt(double num)
```

όπου ο τύπος της παραμέτρου δηλώνεται μέσα στις παρενθέσεις της συνάρτησης.

Σε κάθε εγχειρίδιο της C, καθώς και στη συνέχεια αυτού του βιβλίου, χρησιμοποιούνται αυτοί οι δύο τρόποι για να καθορίσουν τον τρόπο σύνταξης των συναρτήσεων βιβλιοθήκης.

Μετατροπή τύπου (type casting)

Μπορούμε να επιβάλλουμε στο αποτέλεσμα μιας παράστασης να μετατραπεί στον τύπο που επιθυμούμε. Η δυνατότητα αυτή λέγεται **μετατροπή τύπου** (*type casting*) και πολλές φορές είναι όχι μόνο χρήσιμη, αλλά και απαραίτητη.

Η **μετατροπή τύπου** μιας παράστασης γίνεται εφόσον πριν από την παράσταση βάλουμε μέσα σε παρενθέσεις τον τύπο στον οποίο θέλουμε να μετατραπεί.

Ας υποθέσουμε π.χ. ότι οι **a** και **b** είναι μεταβλητές τύπου **int**.

- Η παράσταση **a+b** είναι τύπου **int**.
- Η παράσταση **(float) a+b** είναι τύπου **float**.

Η χρήση μετατροπής τύπου είναι συνήθης σε κάλεσμα συναρτήσεων.

Ας υποθέσουμε ότι έχουμε μια συνάρτηση που ορίζεται ως εξής:

test(float a, float b)

δηλαδή δέχεται δύο παραμέτρους τύπου **float**.

Τη συνάρτηση αυτή θέλουμε να την καλέσουμε από το επόμενο πρόγραμμα, με παραμέτρους τα περιεχόμενα των **x** και **y**.

```
main()
{
    int x=10,y=8;
    test(x,y);
}
```

Αν την καλέσουμε με τον παραπάνω τρόπο, θα έχουμε σίγουρα λάθος αποτελέσματα. Ο λόγος είναι ότι η **test()** περιμένει δύο τιμές τύπου **float** ενώ της παρέχουμε δύο τύπου **int**.

Δεδομένου ότι θέλουμε οι **x** και **y** να είναι οπωσδήποτε τύπου **int**, ο μόνος τρόπος είναι η μετατροπή τύπου.

```
main()
{
    int x=10,y=8;
    test((float)x, (float)y);
}
```

Στο προηγούμενο παράδειγμα, οι τιμές των **x** και **y** μετατρέπονται από τύπου **int** σε τύπου **float** και μεταβιβάζονται στη συνάρτηση **test()** ως παράμετροι τύπου **float**.

Παραδείγματα

- Π.1** Η επόμενη συνάρτηση δέχεται ως παράμετρο έναν αριθμό, τον εμφανίζει αντίστροφα, και επιστρέφει ως τιμή το πλήθος των ψηφίων του.

```
int reverse(int ar)
{
    int y,p,cnt=0;
    do
    {
        y=ar % 10;
        p=ar/10;
        printf("%d",y);
        cnt++;
        ar=p;
    } while (p!=0);
    return cnt;
}
```

Ο αλγόριθμος που ακολουθούμε για το συγκεκριμένο πρόβλημα είναι ο ίδιος με το παράδειγμα Π5 του προηγούμενου κεφαλαίου: Παίρνουμε το υπόλοιπο και το πηλίκο του αριθμού δια 10. Το υπόλοιπο αποτελεί το πρώτο ψηφίο. Αντικαθιστούμε τον αριθμό με το πηλίκο που βρήκαμε και συνεχίζουμε τη διαδικασία μέχρι να βρούμε πηλίκο 0. Η μεταβλητή cnt "μετράει" τα ψηφία του αριθμού. Η συνάρτηση επιστρέφει την τιμή της cnt.

- Π.2** Η επόμενη συνάρτηση δέχεται ως παράμετρο ένα χαρακτήρα και επιστρέφει τιμή 1 αν είναι ελληνικός κεφαλαίος, 2 αν είναι ελληνικός πεζός, 3 αν είναι αριθμητικό ψηφίο (0~9), και 0 σε κάθε άλλη περίπτωση.

```
int isgreek(char ch)
{
    int ap;
    if(ch>='Α' && ch<='Ω') // 'Α'= Ελληνικό άλφα κεφαλαίο!
        ap=1;
    else if(ch>='α' && ch<='ω')
        ap=2;
    else if(ch>='0' && ch<='9')
        ap=3;
}
```

```
    else
        ap=0;
    return ap;
}
```

Π.3 Το επόμενο πρόγραμμα καλεί τη συνάρτηση `cos ()` για να υπολογίσει το συνημίτονο μιας γωνίας.

```
#include <math.h>
#define pi 3.141593
main()
{
    float a;
    double c,rad;
    printf("Δώσε γωνία σε μοίρες:");
    scanf("%f",&a);
    rad=a*pi/180;
    c=cos(rad);
    printf("Το συνημίτονο της γωνίας %f είναι %f\n",a,c);
}
```


Ανασκόπηση Κεφαλαίου 9

- Πέρα από την υποχρεωτική συνάρτηση `main()`, ένα πρόγραμμα μπορεί να περιέχει και άλλες συναρτήσεις με δικά τους ονόματα, οι οποίες μπορεί να ορίζονται μετά ή και πριν από τη `main()`.
- Αν μια συνάρτηση έχει οριστεί μετά από τη `main()` και δεν είναι τύπου `int`, πρέπει να δηλωθεί ξανά πριν από τη `main()`.
- Κάθε συνάρτηση ανήκει σε κάποιον τύπο δεδομένων και μπορεί να επιστρέφει μία τιμή αυτού του τύπου.
- Συναρτήσεις που δεν επιστρέφουν τιμή δηλώνονται ως τύπου `void`.
- Μια συνάρτηση επιστρέφει τιμή μέσω της εντολής `return`.
- Μια συνάρτηση επιστρέφει στο πρόγραμμα που την κάλεσε όταν εκτελεστούν όλες οι προτάσεις της ή όταν εκτελεστεί η εντολή `return`.
- Οι συναρτήσεις μπορεί να έχουν παραμέτρους μέσω των οποίων το πρόγραμμα που τις καλεί, τους μεταβιβάζει δεδομένα.
- Μια συνάρτηση καλείται με το όνομά της και τα ορίσματα των παραμέτρων της, π.χ. `c=add(a,b);`.
- Οι τιμές των ορισμάτων με τα οποία καλείται μια συνάρτηση μεταβιβάζονται ως τιμές στις μεταβλητές των τυπικών παραμέτρων της.
- Τα ορίσματα με τα οποία καλείται μια συνάρτηση πρέπει να είναι του ίδιου τύπου με τον τύπο των αντίστοιχων παραμέτρων της.
- Για να χρησιμοποιήσουμε στον κώδικά μας μια συνάρτηση βιβλιοθήκης αρκεί να συμπεριλάβουμε με `#include` το αρχείο στο οποίο ορίζεται.
- Κατά τη διαδικασία της μεταγλώττισης και στο στάδιο της σύνδεσης, ο κώδικας των συναρτήσεων βιβλιοθήκης που έχουμε χρησιμοποιήσει ενσωματώνεται με τον κώδικα του δικού μας προγράμματος.
- Η τεχνική της μετατροπής τύπου χρησιμοποιείται για τη μετατροπή μιας παράστασης ενός τύπου δεδομένων σε άλλον τύπο δεδομένων, π.χ. `(float)i`.

Ασκήσεις Κεφαλαίου 9

9.1 Βρείτε τα λάθη του επόμενου προγράμματος: ★

```
main()
{
    float x,y;
    printf("*****\n");
    scanf("%f %f",x,y);
    printf("mo=%f\n",mo(x,y));
}

float mo(a,b);
int a,b;
{
    if(a==0 && b==0)
    {
        return 0;
    }
    else
    {
        float mesos;
        mesos=(a+b)/2.0;
    }
    return mesos;
}
```

9.2 Να γραφεί συνάρτηση που να κάνει τα εξής:

- Να δέχεται τρεις παραμέτρους.
- Αν η πρώτη παράμετρος είναι 1, να επιστρέφει ως τιμή το άθροισμα των δύο άλλων παραμέτρων.
- Αν η πρώτη παράμετρος είναι 2, να επιστρέφει ως τιμή το γινόμενο των δύο άλλων παραμέτρων.

- Αν η πρώτη παράμετρος είναι 3, να επιστρέφει ως τιμή το μέσο όρο των δύο άλλων παραμέτρων.
- Αν η πρώτη παράμετρος δεν είναι ούτε 1, ούτε 2, ούτε 3, να εμφανίζει στην οθόνη μήνυμα λάθους "Αντικανονική κλήση συνάρτησης" και να σταματάει το πρόγραμμα με κωδικό εξόδου 1.

Να επιλέξετε εσείς τον τύπο της συνάρτησης και τον τύπο των παραμέτρων της. ★★

- 9.3** Να γραφεί συνάρτηση με όνομα `total()` που να δέχεται ως παράμετρο έναν αριθμό και να επιστρέφει ως τιμή το άθροισμα των αριθμών από το 1 μέχρι την τιμή της παραμέτρου. Για παράδειγμα, η `total(1250)` να επιστρέφει ως τιμή το άθροισμα των αριθμών από το 1 μέχρι το 1250. ★★

- 9.4** Τι κάνει η επόμενη συνάρτηση; ★★

```
int test(ch)
char ch;
{
    char m;
    if(!(ch>='A' && ch<='Ω')) return 0; // 'A'= Ελληνικό κεφαλαίο
    for(m='A';m<=ch;m++)
        putchar(m);
    return 1;
}
```

Τι θα γίνει όταν την καλέσουμε με `test('Θ');`

Τι θα γίνει όταν την καλέσουμε με `test('9');`

Πότε θα επιστρέψει τιμή 0;

- 9.5** Να γραφεί συνάρτηση η οποία θα εμφανίζει δέκα φορές τη φράση "Κατανοήστε τη γλώσσα C". ★

9.6 Ποια από τα παρακάτω αληθεύουν: ★

- ❑ Μια συνάρτηση που δεν επιστρέφει τιμή πρέπει **υποχρεωτικά** να δηλωθεί ως τύπου **void**.
- ❑ Για να χρησιμοποιηθεί μια συνάρτηση βιβλιοθήκης πρέπει στον κώδικα του προγράμματος να συμπεριλάβουμε με **#include** το αρχείο κεφαλίδας στο οποίο δηλώνεται.
- ❑ Αν μια συνάρτηση δεν έχει εντολή **return**, δεν επιστρέφει ποτέ στο πρόγραμμα που την κάλεσε.
- ❑ Όταν καλούμε μια συνάρτηση, ο τύπος των ορισμάτων της πρέπει να είναι αντίστοιχος με τον τύπο των παραμέτρων της.
- ❑ Συναρτήσεις που δεν είναι τύπου **int** και ορίζονται μετά τη **main()** πρέπει να δηλωθούν και πριν από αυτήν (με πρόσθια δήλωση — forward declaration).

9.7 Ο επόμενος τύπος του John Wallis(1616-1703) υπολογίζει προσεγγιστικά το π (3,14...). Να γραφεί πρόγραμμα το οποίο να χρησιμοποιεί αυτή τη σχέση και να υπολογίζει την τιμή του π . Το πρόγραμμα να χρησιμοποιεί συνάρτηση για τον υπολογισμό του δεξιού μέλους της σχέσης. Η συνάρτηση θα δέχεται ως παράμετρο τον τελευταίο όρο του αριθμητή. Να χρησιμοποιηθούν οι 1000 πρώτοι όροι. ★ ★

$$\frac{\pi}{2} = \frac{2^2 \times 4^2 \times 6^2 \times \dots}{3^2 \times 5^2 \times 7^2 \times \dots}$$