

Κεφάλαιο

3

Προετοιμασία για αργότερα



Προετοιμασία για αργότερα

Η C και η μνήμη

Όπως αναφέρθηκε και προηγουμένως, η C χειρίζεται τρεις βασικούς τύπους μεταβλητών: `int`, `float`, και `char`. Φανταζόμαστε την κάθε μεταβλητή σαν ένα κουτί με ένα όνομα και κάποιο περιεχόμενο. Οι μεταβλητές καταλαμβάνουν χώρο στη μνήμη RAM του Η/Υ, αλλά γνωρίζουμε ότι κάθε θέση μνήμης RAM έχει μέγεθος 1 byte.

Κάθε λοιπόν μεταβλητή του προγράμματος καταλαμβάνει ακριβώς τον ίδιο χώρο στη μνήμη του Η/Υ; ΟΧΙ βέβαια.

Κάθε μεταβλητή καταλαμβάνει και έναν αριθμό θέσεων μνήμης (byte), ο οποίος εξαρτάται από τον τύπο της μεταβλητής και από τον τύπο του συστήματος Η/Υ στο οποίο δουλεύουμε.

Στα συστήματα προσωπικών Η/Υ που βασίζονται στους επεξεργαστές 32bit της INTEL, η μεταβλητή τύπου `int` καταλαμβάνει τέσσερα byte, η `float` τέσσερα, και η `char` ένα byte. Αργότερα θα αναφερθούν και άλλοι τύποι μεταβλητών (που στηρίζονται πάντως στους τρεις βασικούς) με διαφορετικά μεγέθη σε byte.

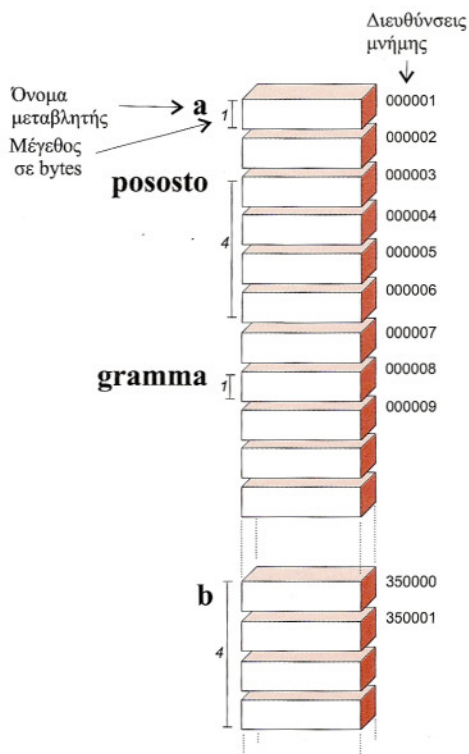
Ας θεωρήσουμε τέσσερις μεταβλητές, οι οποίες δηλώνονται με τις παρακάτω προτάσεις:

```
char a, gramma;
```

```
float pososto;
```

```
int b;
```

Στο διπλανό σχήμα φαίνεται εποπτικά η μνήμη ενός Η/Υ σαν ένα σύνολο από



κουτιά (θέσεις μνήμης) το ένα πάνω από το άλλο. Το κάθε κουτί έχει και έναν αύξοντα αριθμό (ο αριθμός στα δεξιά) ο οποίος αποτελεί τη "**διεύθυνση**" του κουτιού μέσα στη μνήμη. Κάθε μεταβλητή που δηλώνεται, δεσμεύει κάποιες θέσεις μνήμης ανάλογα με τον τύπο της. Έτσι η **a** δεσμεύει μία θέση (την 1), όπως δεσμεύουν και η **gramma** μία θέση (την 8), η **pososto** τέσσερις (τις 3,4,5, και 6), και η **b** επίσης τέσσερις (τις 350000, 350001, 350002, και 350003).

Πέρα λοιπόν από το όνομα και το περιεχόμενο μιας μεταβλητής, έννοιες που είναι ήδη γνωστές, συναντούμε εδώ δύο νέες έννοιες: της **διεύθυνσης** και του **μέγεθους** μιας μεταβλητής.

Διεύθυνση μιας μεταβλητής είναι η διεύθυνση του πρώτου byte των θέσεων (ή της θέσης) μνήμης που δεσμεύει η μεταβλητή.

Έτσι, οι διευθύνσεις των παραπάνω μεταβλητών είναι:

a	→	1
b	→	350000
pososto	→	3
gramma	→	8

Μέγεθος μιας μεταβλητής είναι ο αριθμός των byte που δεσμεύει αυτή η μεταβλητή και καθορίζεται από τον τύπο της.

Η C διαθέτει τελεστές, οι οποίοι επιστρέφουν τη διεύθυνση και το μέγεθος μιας μεταβλητής.

Ο τελεστής &

Ο τελεστής **&** επιστρέφει έναν ακέραιο αριθμό, ο οποίος προσδιορίζει τη διεύθυνση μιας μεταβλητής.

Έτσι, σύμφωνα με το προηγούμενο σχήμα, η παράσταση **&a** επιστρέφει τιμή 1 και η παράσταση **&gramma** τιμή 8.

Όπως θα δούμε αργότερα, αν και οι διευθύνσεις μνήμης είναι στην πράξη ακέραιοι αριθμοί, η C τις χειρίζεται με έναν εντελώς ιδιαίτερο τρόπο.

Ο τελεστής sizeof

Ο τελεστής **sizeof** επιστρέφει το πλήθος των byte που δεσμεύει μία μεταβλητή. Π.χ.

sizeof a → επιστρέφει την τιμή 2

sizeof pososto → επιστρέφει την τιμή 4

12 + sizeof gramma → επιστρέφει την τιμή 13 (12+1).

sizeof(int) → επιστρέφει το 4 (μέγεθος των δεδομένων τύπου **int**)

Τρεις συναρτήσεις παρακαλώ

Στη διδασκαλία της C δεν θα ακολουθήσουμε τον κλασικό τρόπο διδασκαλίας (εντολή προς εντολή) όπως πιθανώς ακολουθείται για τη διδασκαλία άλλων γλωσσών προγραμματισμού. Οι ιδιαιτερότητες της C επιβάλλουν τη γνώση κάποιων βασικών εννοιών και τη χρήση κάποιων εντολών και συναρτήσεων, μόνο και μόνο για να μπορούμε να συνθέσουμε τα πρώτα μας απλά παραδείγματα.

Έτσι χωρίς ακόμα να εμβαθύνουμε και να αναλύσουμε πλήρως το συντακτικό τους, θα γνωρίσουμε τις συναρτήσεις **printf()**, **scanf()**, **exit()**, καθώς και την εντολή **if**.

Η συνάρτηση printf()

Η συνάρτηση **printf()** δίνει τη δυνατότητα για "φορμαρισμένη" εμφάνιση πληροφοριών στην οθόνη του H/Y. Κάνει την ίδια δουλειά όπως οι εντολές εξόδου (στην οθόνη) άλλων γλωσσών προγραμματισμού (π.χ. η εντολή PRINT της BASIC), με διαφορετικό όμως τρόπο. Η σύνταξή της έχει ως εξής:

printf("format string",par1,par2,par3,.....parN);

όπου **format string** ("αλφαριθμητικό μορφοποίησης") είναι ένα σύνολο χαρακτήρων το οποίο περιέχει δύο ειδών πληροφορίες:


- Τους χαρακτήρες που θέλουμε να εμφανίσουμε.
- Ειδικά σύμβολα για την εκτύπωση των τιμών των παραστάσεων που ακολουθούν (**par1~parN**). Τα ειδικά αυτά σύμβολα είναι:

%c για μόνο ένα χαρακτήρα

%d για ακέραιο αριθμό

%f για δεκαδικό αριθμό

και άλλα σύμβολα που θα αναφέρουμε αργότερα.

 Οι παραστάσεις **par1~parN** είναι παραστάσεις των οποίων το αποτέλεσμα θα εμφανιστεί στην οθόνη με τον τρόπο που καθορίζει το αλφαριθμητικό μορφοποίησης (**format string**).

Το παρακάτω πρόγραμμα δείχνει τη χρήση της **printf()**:

```
main()
{
    int a;
    char c;
    float b;
    a=15;
    b=a/2.0;
    c='A';
    printf("a+2=%d b=%f c=%c\n",a+2,b,c);
    printf("Telos\n");
}
```

a+2=17 b=7.5 c=A
telos

Ας αναλύσουμε την πρόταση της **printf()**:

```
printf("a+2=%d b=%f c=%c\n",a+2,b,c);
```

Το αλφαριθμητικό μορφοποίησης περιέχει τρία ειδικά σύμβολα **%d**, **%f**, και **%c**, επομένως πρέπει να ακολουθούν τουλάχιστον τρεις παραστάσεις. Η τιμή της πρώτης θα εμφανιστεί σαν ακέραιος (στη θέση του **%d**), της δεύτερης σαν δεκαδικός (στη θέση του **%f**), και της τρίτης σαν χαρακτήρας (στη θέση του **%c**). **Οτιδήποτε άλλο υπάρχει μέσα στο αλφαριθμητικό μορφοποίησης θα εμφανίζεται στην οθόνη όπως ακριβώς είναι.**

Ο χαρακτήρας `\n` στο τέλος του αλφαριθμητικού μορφοποίησης επιβάλλει την αλλαγή γραμμής μετά το τέλος της εμφάνισης των χαρακτήρων.

Η επόμενη `printf()` του προγράμματος, `printf("Τελος\n")`, δεν έχει παραστάσεις να ακολουθούν και το αλφαριθμητικό μορφοποίησης "Τελος\n", δεν περιέχει κανένα ειδικό σύμβολο (από τα `%d`, `%f` & `%c`). Το μόνο που κάνει η συγκεκριμένη `printf()` είναι να εμφανίζει αυτούσια τα περιεχόμενα του αλφαριθμητικού μορφοποίησης.


Το `\n` είναι ένας χαρακτήρας (και όχι δύο ξεχωριστοί) από τους λεγόμενους "χαρακτήρες διαφυγής" (escape characters), οι οποίοι δεν εμφανίζονται στην οθόνη αλλά ελέγχουν μία συγκεκριμένη λειτουργία της.

Οι χαρακτήρες διαφυγής στη C είναι:

- `\n` new line (αλλαγή γραμμής)
- `\b` backspace (μία θέση πίσω)
- `\f` form feed (νέα σελίδα)
- `\r` carriage return (αρχή γραμμής)
- `\t` tab (επόμενη στηλοθετημένη στήλη)
- `\0` null character (byte 00000000)

Οι χαρακτήρες αυτοί μπορούν να χρησιμοποιηθούν είτε μόνοι τους ("`\n`") είτε σε σύνολα χαρακτήρων (character strings) π.χ. "`\f Κορυφή σελίδας\n`".

Αν το αλφαριθμητικό μορφοποίησης σε μία `printf()` δεν τελειώνει με το χαρακτήρα αλλαγής γραμμής `\n`, τότε η επόμενη εκτύπωση αρχίζει από τη θέση που σταμάτησε η `printf()`.

-  Η `printf()` μπορεί να χρησιμοποιηθεί μόνο με το αλφαριθμητικό μορφοποίησης, οπότε απλώς εμφανίζει τους χαρακτήρες που περιέχει. Για παράδειγμα, η:
`printf("C is the best");`
θα εμφανίσει τους χαρακτήρες
C is the best

Η συνάρτηση scanf()

Η συνάρτηση `scanf()` χρησιμοποιείται για την είσοδο δεδομένων από το πληκτρολόγιο (όπως π.χ. η εντολή INPUT στη BASIC). Η σύνταξή της είναι σχεδόν ίδια με της `printf()`:

`scanf("format string",addr1,addr2,addr3,.....addrN);` όπου,

το **format string** (αλφαριθμητικό μορφοποίησης) περιέχει τριών ειδών πληροφορίες:

- **Ειδικά σύμβολα** για την **ανάγνωση** των τιμών που θα ανατεθούν στις μεταβλητές, με τις **διευθύνσεις** που ακολουθούν (`addr1~addrN`).
`%c` για έναν απλό χαρακτήρα
`%d` για ακέραιο αριθμό
`%f` για δεκαδικό αριθμό
 και άλλα σύμβολα που θα αναφερθούν αργότερα.
- **Χαρακτήρες διαστήματος.** Αναγκάζουν τη `scanf()` να διαβάσει και να αγνοήσει ένα ή περισσότερα κενά διαστήματα ή χαρακτήρες αλλαγής γραμμής, στη σειρά των δεδομένων που πληκτρολογούνται. Για παράδειγμα, το αλφαριθμητικό μορφοποίησης:
`"%d %d"`
 διαβάζει δύο ακέραιους, αγνοώντας τα μεταξύ τους κενά και τους χαρακτήρες αλλαγής γραμμής (αν υπάρχουν).
- **Άλλους χαρακτήρες.** Αναγκάζουν τη `scanf()` να διαβάσει και να αγνοήσει ένα συγκεκριμένο χαρακτήρα. Αν ο χαρακτήρας δεν βρεθεί, η `scanf()` τερματίζει. Για παράδειγμα, το αλφαριθμητικό μορφοποίησης:
`"%d,%d"`
 διαβάζει έναν ακέραιο, μετά διαβάζει και αγνοεί ένα κόμμα (,) και τέλος διαβάζει ακόμη έναν ακέραιο. Αν δεν βρεθεί το κόμμα αμέσως μετά από τον πρώτο ακέραιο, τότε η `scanf()` τερματίζει χωρίς να διαβάσει τον επόμενο αριθμό.

Οι παραστάσεις `addr1~addrN` είναι παραστάσεις των οποίων το αποτέλεσμα αποτελεί τη διεύθυνση μνήμης μιας μεταβλητής. Τα δεδομένα που πληκτρολογούνται θα καταχωριστούν στις μεταβλητές με αυτές τις διευθύνσεις.

Ο τελεστής & χρησιμοποιείται συνήθως σε μία συνάρτηση `scanf()` για να αποδώσει τη διεύθυνση μιας μεταβλητής: Για παράδειγμα, η

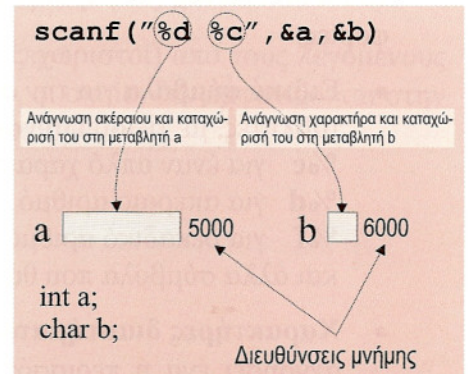
```
scanf("%d %c", &a, &b);
```

περιμένει να πληκτρολογηθούν ένας ακέραιος αριθμός και ένας χαρακτήρας. Ο αριθμός θα καταχωριστεί στη μεταβλητή `a` (με διεύθυνση `&a`) ενώ ο χαρακτήρας στη μεταβλητή `b` (με διεύθυνση `&b`). Φυσικά, οι μεταβλητές `a` και `b` πρέπει να είναι τύπου `int` και `char` αντίστοιχα. Κατά την πληκτρολόγηση, πρέπει να υπάρχουν μεταξύ του αριθμού και του χαρακτήρα ένα ή περισσότερα διαστήματα ή αλλαγές γραμμής, τα οποία αγνοούνται.

Για να δώσουμε στο χρήστη να καταλάβει τι πρόκειται να πληκτρολογήσει, χρησιμοποιούμε πριν από τη `scanf()` μία `printf()` ή κάποια άλλη συνάρτηση εξόδου (π.χ. την `puts()` που θα συναντήσουμε αργότερα), με την οποία εμφανίζουμε το μήνυμα.

Το επόμενο παράδειγμα δείχνει τη χρήση της `scanf()` μέσα σε ένα πρόγραμμα:

```
main()
{
    int a,b;
    float c;
    printf("δώσε 2 αριθμούς:");
    scanf("%d %d", &a, &b);
    //υπολογισμός μέσου όρου
    c=(a+b)/2.0;
    //εμφάνιση αποτελεσμάτων
    printf("ο μέσος όρος των %d και %d είναι %f",a,b,c);
}
```



Η παρουσία άλλων χαρακτήρων μεταξύ των ειδικών χαρακτήρων, έχει σαν αποτέλεσμα η `scanf()` να περιμένει να διαβάσει και να αγνοήσει τους συγκεκριμένους χαρακτήρες.

Για παράδειγμα, η συνάρτηση:

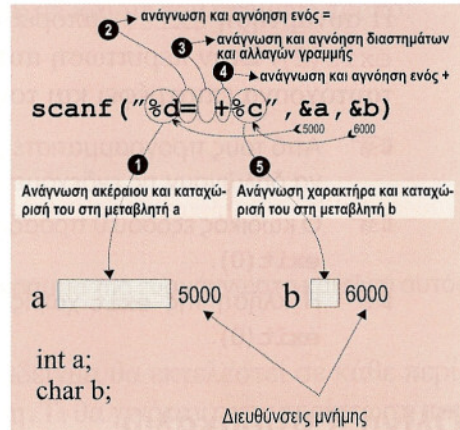
```
scanf("%d= +%c", &a, &b);
```

A. Θα διαβάσει έναν ακέραιο αριθμό και θα τον καταχωρίσει στη μεταβλητή **a**.

B. Θα διαβάσει και θα αγνοήσει ένα `=`, ένα ή περισσότερα διαστήματα ή αλλαγές γραμμής, και ένα `+`.

Γ. Θα διαβάσει ένα χαρακτήρα και θα τον καταχωρίσει στη μεταβλητή **b**.

☞ Αν οι ενδιάμεσοι χαρακτήρες δεν πληκτρολογηθούν με τη σειρά που αναφέραμε, η `scanf()` θα τερματίσει χωρίς να συνεχίσει την ανάγνωση δεδομένων από το πληκτρολόγιο.



Η συνάρτηση `exit()`

Η συνάρτηση `exit()` έχει αποτέλεσμα τον άμεσο τερματισμό του προγράμματος. Για παράδειγμα, στο επόμενο πρόγραμμα:

```
main()
{
    int a;
    char c;
    printf("Αυτή η πρόταση θα εκτελεστεί");
    exit();
    printf("Αυτή δεν θα εκτελεστεί ποτέ");
}
```

η τελευταία `printf()` δεν θα εκτελεστεί ποτέ γιατί με την κλήση της συνάρτησης `exit()` το πρόγραμμα θα τερματιστεί νωρίτερα.

Η συνάρτηση `exit()` μπορεί να χρησιμοποιηθεί και με μία παράμετρο, π.χ. `exit(2)`. Στην περίπτωση αυτή, το πρόγραμμα τερματίζεται και πάλι, αλλά ταυτόχρονα επιστρέφει και τον κωδικό εξόδου 2 στο λειτουργικό σύστημα.

- 👉 Από τους προγραμματιστές χρησιμοποιούνται διαφορετικοί κωδικοί εξόδου για να δηλώνουν τις ενδεχόμενες περιπτώσεις τερματισμού ενός προγράμματος.
- 👉 Ο κωδικός εξόδου 0 προσδιορίζει κανονικό τερματισμό του προγράμματος, π.χ. `exit(0)`.
- 👉 Η κλήση της `exit` χωρίς παράμετρο, ως `exit()`, είναι ισοδύναμη με την `exit(0)`.

Ολίγη if παρακαλώ

Στο σημείο αυτό θα αναφέρουμε την εντολή `if`, μόνο και μόνο επειδή θα την χρησιμοποιήσουμε σε επόμενα παραδείγματα για την κατανόηση άλλων εννοιών. Θα αναφέρουμε λίγες μόνο από τις δυνατότητες της `if`, και μάλιστα μέσα από παραδείγματα, δεδομένου ότι το πλήρες συντακτικό και η ανάλυση της εντολής θα γίνουν σε επόμενο κεφάλαιο. Το συντακτικό της `if` έχει ως εξής:

`if (λογική παράσταση) πρόταση-A; [else πρόταση-Ψ;]`

Αν η λογική παράσταση είναι αλήθεια, εκτελείται η **πρόταση-A**, και αν είναι ψέμα, η **πρόταση-Ψ** (αν υπάρχει). Το `else` και η πρόταση-Ψ είναι προαιρετικά. Οι αγκύλες `[]` δεν μετέχουν στο συντακτικό της εντολής, αλλά δείχνουν ότι το τμήμα αυτό της `if` είναι προαιρετικό.

Θα χρησιμοποιούμε τις αγκύλες `[]` από τώρα και στο εξής για να σηματοδοτούμε τα προαιρετικά τμήματα μιας εντολής. Οι αγκύλες δεν πρέπει να συγχέονται με τα άγκιστρα `{ }` που σηματοδοτούν την αρχή και το τέλος μιας σύνθετης πρότασης.

- 👉 Στη C αληθής θεωρείται κάθε παράσταση (λογική ή αριθμητική) που επιστρέφει τιμή διάφορη του μηδενός. Για παράδειγμα, η παράσταση `5+6` θεωρείται αληθής ενώ η `6-6` θεωρείται ψευδής.


Για παράδειγμα: η επόμενη εντολή `if` ελέγχει την τιμή του `x`. Αν είναι μεγαλύτερη από 5 τότε εμφανίζει στην οθόνη `"x>5"`, διαφορετικά εμφανίζει `"x<=5"`.

```
if (x>5) printf("x>5\n"); else printf("x<=5\n");
```

Η παραπάνω εντολή θα μπορούσε να γραφτεί και με τον επόμενο τρόπο:

```
if (x>5)
    printf("x>5\n");
else
    printf("x<=5\n");
```

χωρίς να αλλάξει καθόλου το νόημά της.

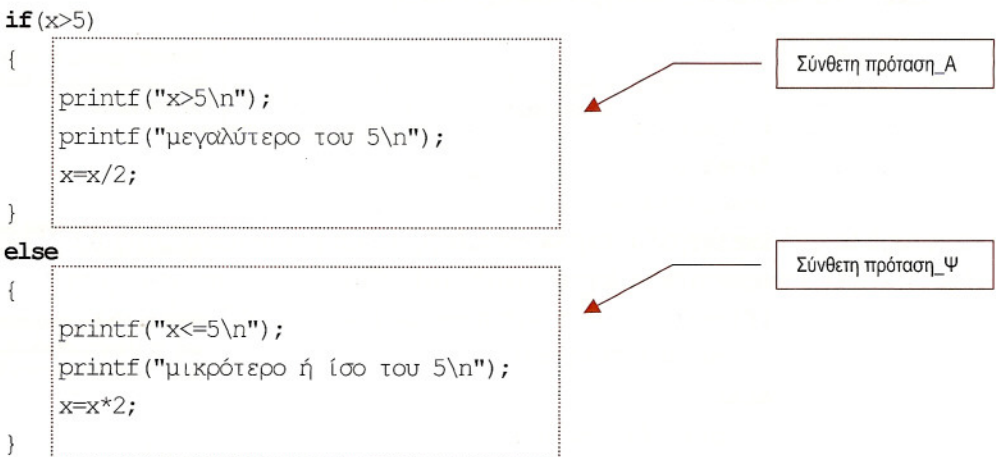
 Αυτός ο δεύτερος τρόπος κάνει το πρόγραμμα πιο ευανάγνωστο και είναι αυτός που θα ακολουθήσουμε στη συνέχεια.

Παρατηρούμε ότι στο προηγούμενο παράδειγμα θα εκτελεστεί σε κάθε περίπτωση (αλήθεια ή ψέμα) μία απλή πρόταση. Τι θα γινόταν στην περίπτωση που θα θέλαμε να εκτελεστούν περισσότερες προτάσεις;

Αναφερθήκαμε προηγουμένως στην έννοια της σύνθετης πρότασης (compound statement) και την ικανότητα της C να τη μεταχειρίζεται σαν οποιαδήποτε απλή πρόταση. Επομένως, μπορούμε ανά πάσα στιγμή να χρησιμοποιήσουμε στη θέση μιας απλής πρότασης μια σύνθετη πρόταση, η οποία μπορεί να συνίσταται από πολλές απλές προτάσεις.

Το επόμενο παράδειγμα δείχνει τη χρήση δύο σύνθετων προτάσεων μέσα σε μια εντολή `if`.

```
if (x>5)
{
    printf("x>5\n");
    printf("μεγαλύτερο του 5\n");
    x=x/2;
}
else
{
    printf("x<=5\n");
    printf("μικρότερο ή ίσο του 5\n");
    x=x*2;
}
```



Αν η τιμή του `x` είναι μεγαλύτερη από 5, τότε εκτελείται η σύνθετη **πρόταση_Α**, διαφορετικά η σύνθετη **πρόταση_Ψ**.

Παραδείγματα

- Π.1** Το παρακάτω πρόγραμμα ζητάει τρεις βαθμούς ενός μαθητή και υπολογίζει το μέσο όρο τους. Αν ο μέσος όρος είναι μεγαλύτερος ή ίσος με 10, εμφανίζει "Πέρασες με βαθμό ##", διαφορετικά εμφανίζει "Κόπηκες" (στη θέση του ## εμφανίζεται ο μέσος όρος).

```
main()
{
    int b1,b2,b3;
    float mo;
    printf("Δώσε τρεις βαθμούς:");
    scanf("%d %d %d",&b1,&b2,&b3);
    mo=(b1+b2+b3)/3.0;
    if(mo>=10)
        printf("Πέρασες με βαθμό %f\n",mo);
    else
        printf("Κόπηκες\n");
}
```

- Π.2** Το παρακάτω πρόγραμμα έχει λάθη:

```
main()
{
    int a,b,c;
    float d;
    scanf("%d %d",a,b);
    int k;
    scanf("%c",&c);
    if(c==1)
        printf("NAI\n");
    else
        printf("OXI\n");
}
```

Η scanf() χρειάζεται τις διευθύνσεις των μεταβλητών. Θα έπρεπε να ήταν &a,&b

Δεν μπορεί να υπάρχει δηλωτική πρόταση μετά από εκτελέσιμες.

Εφόσον ζητείται ακέραιος αριθμός, το αλφαριθμητικό μορφοποίησης θα έπρεπε να είναι %d.

Π.3 Το επόμενο πρόγραμμα ζητάει έναν αριθμό και απαντάει αν είναι ζυγός ή μονός:

```
main()
{
    int a,p;
    scanf("%d",&a);
    p=a/2;
    if(a==p*2)
        printf("ζυγός");
    else
        printf("μονός");
}
```

Η scanf() ζητάει έναν αριθμό τον οποίο καταχωρίζει στη μεταβλητή a.

Η μεταβλητή p παίρνει ως τιμή το ακέραιο μισό της a. Αν το a είναι 5 το p θα πάρει την τιμή 2 και όχι 2.5 (το 0.5 χάνεται διότι η p είναι τύπου int).

Αν το διπλάσιο του p είναι ίσο με το a, αυτό σημαίνει ότι ο a είναι ζυγός, διαφορετικά ότι είναι μονός.

Π.4 Το επόμενο πρόγραμμα εμφανίζει στην οθόνη τη λέξη "ΝΑΙ" και ως τιμή του a το 5. Το πρόγραμμα περιέχει ένα σύννηθες λάθος στη χρήση της if.

```
main()
{
    int a;
    a=2
    if(a=5)
        printf("ΝΑΙ");
    else
        printf("ΟΧΙ");
    printf("a=%d\n",a);
}
```

Η παράσταση a=5 δεν είναι λογική παράσταση, αλλά αναθέτει στο a το 5 και έχει αποτέλεσμα 5 (αληθές). Αν θέλαμε να γίνει σύγκριση του a με το 5 θα έπρεπε να χρησιμοποιήσουμε δύο ίσον (a==5). Η if θεωρεί αληθή την παράσταση a=5 (εφόσον το αποτέλεσμα της είναι 5) οπότε εμφανίζει το "ΝΑΙ".

Η μεταβλητή a περιέχει το 5 όπως της ανατέθηκε από την παράσταση a=5 της if.

Π.5 Το παρακάτω πρόγραμμα ζητάει να πληκτρολογηθούν δύο ακέραιοι αριθμοί οι οποίοι να χωρίζονται υποχρεωτικά με ένα κόμμα (,) και εμφανίζει το μέσο όρο τους και το άθροισμά τους.

```
main()
{
    int a,b,s;
    float mo;
```

```
scanf("%d,%d",&a,&b);  
mo=(a+b)/2.0;  
s=a+b;  
printf("MO=%f\n",mo);  
printf("Αθροισμα=%d\n",s);  
}
```

Το κόμμα (,) μεταξύ των %d αναγκάζει τη scanf() να αναζητήσει μεταξύ των δύο αριθμών που θα πληκτρολογηθούν υποχρεωτικά ένα κόμμα. Σε διαφορετική περίπτωση η scanf() τερματίζει χωρίς να διαβάσει όλες της τις τιμές.

Ανασκόπηση Κεφαλαίου 3

- Κάθε μεταβλητή στη C χαρακτηρίζεται από το όνομά της, το μέγεθός της, και τη διεύθυνσή της.
- Ανάλογα με τον τύπο της, η κάθε μεταβλητή καταλαμβάνει συγκεκριμένο αριθμό από byte.
- Ο τελεστής & αποδίδει τη διεύθυνση μιας μεταβλητής, ενώ ο τελεστής sizeof το μέγεθός της.
- Η συνάρτηση printf() χρησιμοποιείται για την εμφάνιση πληροφοριών στην οθόνη.
- Η συνάρτηση scanf() χρησιμοποιείται για την ανάγνωση δεδομένων (χαρακτήρων και αριθμών) από το πληκτρολόγιο.
- Η συνάρτηση scanf() χρειάζεται τις διευθύνσεις των μεταβλητών στις οποίες θα καταχωρίσει τα δεδομένα που πληκτρολογούνται.
- Η εντολή if ελέγχει μία λογική παράσταση και, ανάλογα με το αποτέλεσμα, εκτελεί μία ή περισσότερες προτάσεις.

Ασκήσεις Κεφαλαίου 3

- 3.1** Να γραφεί ένα πρόγραμμα το οποίο να ζητάει τρεις δεκαδικούς αριθμούς, να υπολογίζει, και να εμφανίζει τον μέσο όρο τους. ★★

- 3.2** Τι αποτέλεσμα θα έχει το επόμενο πρόγραμμα; ★

```
main()
{
    int a=4,b=5;
    char ch;
    ch='A';
    printf("%d %d %c",a,b,ch);
    printf("%d %d %d\n",a,b,ch);
    printf("%d\n%d \n%c\n",a,b,ch);
    printf("telos");
}
```

- 3.3** Να γραφεί πρόγραμμα το οποίο να ζητάει 3 αριθμούς και να υπολογίζει το άθροισμα, το γινόμενο, και το μέσο όρο τους. Το πρόγραμμα να μας βγάζει μηνύματα για το τι πρέπει να δώσουμε και να βγάζει τα αποτελέσματα όπως στο διπλανό παράδειγμα. ★★★

Δώσε τον πρώτο αριθμό: 6
 Δώσε το δεύτερο αριθμό: 2
 Δώσε τον τρίτο αριθμό: 10
 Το άθροισμα των 6,2,10 είναι 18
 Το γινόμενο των 6,2,10 είναι 120
 Ο μέσος όρος των 6,2,10 είναι 6

- 3.4** Τι αποτέλεσμα θα έχει το παρακάτω πρόγραμμα; ★★

```
main()
{
    int a,b;
    float f;
    char ch;
    printf("%d %d %d\n",sizeof a,sizeof f,sizeof ch);
}
```

```
scanf("%d %f %c",&a,&f,&ch);  
printf("%d %f %c\n",a,f,ch);  
}
```

3.5 Εξηγήστε τη λειτουργία του επόμενου προγράμματος: ★★

```
main()  
{  
    int a,b;  
    scanf("%d %d",&a,&b);  
    if(a>b)  
        printf("%d",a);  
    else  
        printf("%d",b);  
}
```

3.6 Να γραφεί πρόγραμμα το οποίο να ζητάει τρεις ακέραιους αριθμούς και να εμφανίζει το μεγαλύτερο από αυτούς. ★★★

3.7 Ποια από τα παρακάτω αληθεύουν: ★

- ☐ Η `scanf()` χρειάζεται τις διευθύνσεις των μεταβλητών στις οποίες θα καταχωρίσει τα δεδομένα που θα πληκτρολογήσουμε.
- ☐ Το μέγεθος ενός τύπου δεδομένων στη C είναι πάντα το ίδιο και ανεξάρτητο από το σύστημα στο οποίο δουλεύουμε.
- ☐ Ο έλεγχος `if (a=5)` είναι πάντα αληθής.
- ☐ Αν μέσα σε ένα πρόγραμμα δεν υπάρχει κλήση της `exit()`, το πρόγραμμα δεν θα τερματιστεί ποτέ.
- ☐ Ο τελεστής `sizeof` μπορεί να εφαρμοστεί και σε τύπο δεδομένων π.χ. `sizeof(char)`.

3.8 Να γραφεί πρόγραμμα το οποίο να ζητάει να πληκτρολογήσουμε δύο ακέραιους αριθμούς που θα χωρίζονται με ένα κόμμα (,) και ένα αστεράκι (*) και να εμφανίζει το άθροισμά τους. ★★

12,*34