

Κεφάλαιο

2

Μια πρώτη ματιά στη C



Μια πρώτη ματιά στη C

Στο κεφάλαιο αυτό ξεκινά μια μικρή "περιοδεία" στη γλώσσα C. Γίνεται μια πρώτη γνωριμία, ώστε να αποκτήσει ο αναγνώστης μια σφαιρική εικόνα της δομής και των χαρακτηριστικών της γλώσσας.

Τα περισσότερα από όσα αναφέρονται στο παρόν κεφάλαιο θα εξεταστούν ξανά και πιο αναλυτικά στα επόμενα κεφάλαια.

Η δομή ενός προγράμματος στη C

Ένα απλό πρόγραμμα στη C αποτελείται συνήθως από 3 βασικά τμήματα:

Μία συνάρτηση με όνομα `main()`: Αυτή η συνάρτηση είναι αυτή που καλείται και εκτελείται πρώτη. Κάθε πρόγραμμα πρέπει να έχει **μία και μόνο μία** συνάρτηση `main()`. Η συνάρτηση `main()`, όπως και κάθε συνάρτηση της C, έχει την παρακάτω μορφή:

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

Δηλώσεις μεταβλητών: Όλες οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα της C πρέπει να έχουν προηγουμένως δηλωθεί. Μία δήλωση μεταβλητής περιλαμβάνει το όνομα της μεταβλητής και τον τύπο της (μπορεί επίσης να περιλαμβάνει και την αρχική της τιμή). Έτσι, η πρόταση


`int a,b;`

δηλώνει δύο ακέραιες μεταβλητές με ονόματα **`a`** και **`b`**.

 Κάθε δηλωτική πρόταση, τερματίζεται με ελληνικό ερωτηματικό (;)

Οι διάφοροι τύποι μεταβλητών θα συζητηθούν στο επόμενο κεφάλαιο.

Εκτελέσιμες προτάσεις: Μετά από τις δηλώσεις μεταβλητών πρέπει να ακολουθούν οι εκτελέσιμες προτάσεις (εντολές ή παραστάσεις) του προγράμματος.

 Κάθε εκτελέσιμη πρόταση τερματίζεται με ελληνικό ερωτηματικό (;).

Ένα πιο σύνθετο πρόγραμμα περιλαμβάνει περισσότερες συναρτήσεις πέρα από τη συνάρτηση `main()`:

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

function1()
{
    δηλώσεις μεταβλητών της συνάρτησης function1();
    εκτελέσιμες προτάσεις της συνάρτησης function1();
}

function2()
{
    δηλώσεις μεταβλητών της συνάρτησης function2();
    εκτελέσιμες προτάσεις της συνάρτησης function2();
}
```

Παρατηρούμε ότι μπορούμε να δηλώνουμε ξεχωριστές μεταβλητές σε κάθε συνάρτηση. Οι μεταβλητές αυτές καλούνται τοπικές μεταβλητές (local variables) και μπορούν να χρησιμοποιηθούν μόνο μέσα στις συναρτήσεις στις οποίες δηλώθηκαν. Στο Κεφάλαιο 10 αναλύονται σε βάθος η χρήση και οι ιδιότητες των τοπικών μεταβλητών.

Σχόλια προγράμματος

Ο σωστός σχεδιασμός ενός προγράμματος επιβάλλει τη χρήση σχολίων μέσα στο ίδιο το πρόγραμμα με σκοπό να το κάνει περισσότερο ευανάγνωστο και κατανοητό. **Τα σχόλια μπορούν να περιλαμβάνουν οποιοδήποτε μήνυμα, αρχίζουν με το ζεύγος των χαρακτήρων `/*` και τελειώνουν με τους χαρακτήρες `*/`.**

Ακολουθεί ένα παράδειγμα μιας συνάρτησης `main()` με σχόλια.


```
/* Το πρώτο μου πρόγραμμα στη C
Μυτιλήνη
24/10/1983 */

main()
{
    /*η χρήση της συνάρτησης printf()*/
    printf("Hello world\n");
    /*Εμφανίζει ένα σύνολο χαρακτήρων*/
}
```

Παρατηρούμε ότι σχόλια μπορούν να μπουν οπουδήποτε μέσα σε ένα πρόγραμμα της C, είτε στην αρχή μιας συνάρτησης είτε ενδιάμεσα στον κώδικα.

Αρκετοί μεταγλωττιστές της C και της C++, μεταξύ των οποίων και ο DEV C/C++, αναγνωρίζουν "σχόλια γραμμής" που ξεκινάνε με δύο συνεχόμενες καθετούς (//).

Με αυτόν τον τρόπο κάθε γραμμή σχολίων πρέπει να αρχίζει από //.

Για παράδειγμα:

```
// Το πρώτο μου πρόγραμμα στη C
// Μυτιλήνη
// 24/10/1983

main()
{
    //η χρήση της συνάρτησης printf()
    printf("Hello world\n"); //θα εμφανίσει το κείμενο
}
```

Όταν ο μεταγλωττιστής (compiler) της C εντοπίσει το συνδυασμό /* αγνοεί οτιδήποτε βρει μέσα στο πρόγραμμα μέχρι το επόμενο */.

Αρκετοί μεταγλωττιστές αγνοούν επίσης και οτιδήποτε υπάρχει μετά από διπλή καθετο (//) μέχρι το τέλος της γραμμής.

Δηλώσεις μεταβλητών

Πριν χρησιμοποιηθεί οποιαδήποτε μεταβλητή, πρέπει πρώτα να έχει δηλωθεί. Στην πρόταση της δήλωσης ενημερώνουμε το μεταγλωττιστή της C για το **όνομα** και τον **τύπο** της μεταβλητής. Η σύνταξη μιας δηλωτικής πρότασης στη C έχει την παρακάτω μορφή:

type **ονομα1,ονομα2, ... ονομαN;**

όπου **type** ο τύπος δεδομένων των μεταβλητών και **ονομα1, ονομα2, ... ονομαN** είναι ονόματα μεταβλητών. Οι βασικοί τύποι μεταβλητών στη C είναι τρεις:

int ακέραιος αριθμός

char χαρακτήρας

float αριθμός κινητής υποδιαστολής (με δεκαδικά ψηφία)

Πέρα από τους παραπάνω τύπους, υπάρχουν και άλλοι τύποι μεταβλητών, που θα αναλυθούν αργότερα και οι οποίοι στηρίζονται σε αυτούς τους τρεις βασικούς τύπους.

Το όνομα μιας μεταβλητής μπορεί να αποτελείται από χαρακτήρες, αριθμούς, και το χαρακτήρα υπογράμμισης "_". Ο πρώτος χαρακτήρας του ονόματος πρέπει να είναι γράμμα. Το πλήθος των χαρακτήρων του ονόματος θεωρητικά είναι απεριόριστο αλλά μπορεί να μεταβάλλεται από έκδοση σε έκδοση².

Δεν επιτρέπεται η χρήση ελληνικών χαρακτήρων σε ονόματα μεταβλητών. Έτσι, μπορούμε π.χ. να έχουμε μεταβλητές με τα ονόματα:

```
a
teliko_apotelesma
a23
```

αλλά όχι με τα ονόματα:

```
Τελικό_σύνολο            // ❌ Ελληνικοί χαρακτήρες
```

² Οι ίδιοι περιορισμοί ισχύουν και για όλα τα άλλα αναγνωριστικά που ορίζονται από τον χρήστη, όπως ονόματα συναρτήσεων, τύπων κ.λπ., τα οποία θα συναντήσουμε αργότερα.

23a // < Εκκινάει από αριθμό
number of students // < Περιέχει κενά διαστήματα

Οι παρακάτω δηλωτικές προτάσεις αντιστοιχούν σε τέσσερις μεταβλητές διαφορετικών τύπων:

int number_of_students, a2;


Με την πρόταση αυτή δηλώνουμε δύο ακέραιες μεταβλητές με ονόματα `number_of_students` και `a2`.

char apantisi;

Με την πρόταση αυτή δηλώνουμε μία μεταβλητή χαρακτήρα με όνομα `apantisi`.

float mesos_oros;

Με την πρόταση αυτή δηλώνουμε μία μεταβλητή για δεκαδικό αριθμό με όνομα `mesos_oros`.

 **Στη C οι πεζοί και οι κεφαλαίοι χαρακτήρες είναι διακριτοί.** Αυτό σημαίνει ότι η μεταβλητή `a` είναι διαφορετική από την `A`. Επίσης, όλες οι εντολές και συναρτήσεις της C συντάσσονται με πεζούς χαρακτήρες.

Αρχικές τιμές μεταβλητών

Στην πρόταση δήλωσης μιας μεταβλητής είναι δυνατόν ταυτόχρονα να της αναθέσουμε και μία τιμή, π.χ.

int a=4,b;

float c=4.6,m;

char ch='A';

Οι παραπάνω προτάσεις είναι δηλωτικές για τις μεταβλητές `a`, `b`, `c`, `m` και `ch` και ταυτόχρονα δίνουν στις μεταβλητές `a`, `c` και `ch` τις τιμές 4, 4.6, και 'A' αντίστοιχα.


Μεταβλητές μόνο για ανάγνωση

Στη C μπορούμε να δηλώσουμε μεταβλητές στις οποίες να δώσουμε αρχική τιμή χωρίς όμως στη συνέχεια να μπορούμε να μεταβάλλουμε το περιεχόμενό τους. Μια τέτοια μεταβλητή δηλώνεται όπως και οι υπόλοιπες, προσθέτοντας μπροστά από τον τύπο της το διακριτικό **const**.

```
const int a=8;
const float c=4.6;
```

Οι παραπάνω προτάσεις δηλώνουν δύο μεταβλητές μόνο για ανάγνωση **a** και **c**, με αρχικές τιμές 8 και 4.6 αντίστοιχα. Στις μεταβλητές αυτές δεν επιτρέπεται να αναθέσουμε αργότερα άλλες τιμές.

Η χρήση τέτοιου είδους μεταβλητών είναι πολύ περιορισμένη.

 ΠΡΟΣΟΧΗ όταν δηλώνουμε μεταβλητές μόνο για ανάγνωση πρέπει απαραίτητα να τους δίνουμε αρχική τιμή.

Προτάσεις

Στη C μπορεί να έχουμε δύο ειδών προτάσεις:

- Τις προτάσεις δήλωσης
- Τις εκτελέσιμες προτάσεις (εντολές ή παραστάσεις)

Οι προτάσεις δήλωσης, πέρα από αυτές που δηλώνουν μεταβλητές, όπως αναφέραμε προηγουμένως, μπορεί να δηλώνουν συναρτήσεις ή μεταβλητές κάποιου σύνθετου τύπου. Προς το παρόν, μέχρι να φτάσουμε στο σημείο να αναφερθούμε σε αυτές τις διαφορετικές προτάσεις δήλωσης, θα έχουμε υπόψη μόνο τις προτάσεις που δηλώνουν μεταβλητές των τριών βασικών τύπων (**int**, **char**, και **float**).

Οι εκτελέσιμες προτάσεις είναι εντολές ή παραστάσεις που εκτελούν κάποια συγκεκριμένη λειτουργία. Στο παρακάτω παράδειγμα,

```
main()
{
    int x,y;    // Δηλωτική πρόταση
    x=5;        // Εκτελέσιμη πρόταση
```

```
y=x+4*20; // < Εκτελέσιμη πρόταση  
}
```

η πρώτη πρόταση είναι δηλωτική και οι δύο επόμενες εκτελέσιμες.

Παρατηρούμε ότι στο συγκεκριμένο παράδειγμα οι εκτελέσιμες προτάσεις μπορεί να αποτελούνται από απλές παραστάσεις, οι οποίες συνθέτουν άλλες πιο πολύπλοκες παραστάσεις. Στην επόμενη παράγραφο αναλύεται η φιλοσοφία των παραστάσεων στη C.

Παραστάσεις

Μια παράσταση αποτελείται από σταθερές, μεταβλητές, και συναρτήσεις, συνήθως σε συνδυασμό με αριθμητικούς (π.χ. +, -, /) ή λογικούς τελεστές (π.χ. <, <=, ==).

Σταθερές (constants): Μια σταθερά μπορεί να είναι του τύπου `int`, `char`, ή `float` και συνήθως δεν αποτελεί από μόνη της μια πρόταση σε ένα πρόγραμμα της C (αν και συντακτικά αυτό είναι δυνατόν, όπως θα αναφέρουμε και παρακάτω).

Παραδείγματα σταθερών είναι:

```
7          σταθερά τύπου int  
123.4      σταθερά τύπου float  
'A'       σταθερά τύπου char
```

Όπως κάθε παράσταση, έτσι και οι σταθερές έχουν μία τιμή και έναν τύπο. Π.χ. η σταθερά 123.4 έχει τιμή 123.4 και τύπο `float`.

Μεταβλητές (variables): Το όνομα μιας μεταβλητής αποτελεί από μόνο του μια απλή παράσταση. Παραστάσεις με μία μεταβλητή συνήθως δεν αποτελούν από μόνες τους μια πρόταση αλλά συνήθως συνδυάζονται με άλλες παραστάσεις. Ακολουθούν παραδείγματα χρήσης μεταβλητών σε παραστάσεις:

Μεταβλητή ως απλή παράσταση:

```
a;
```



Μεταβλητή ως τμήμα μιας σύνθετης παράστασης:

$$a + 6;$$

Η τιμή της παραπάνω παράστασης υπολογίζεται ως το άθροισμα του περιεχομένου της μεταβλητής a και του 6.

$$b = a + 6;$$

Σε αυτή την περίπτωση, ο τελεστής ίσον (=) καταχωρίζει το αποτέλεσμα της παράστασης $a + 6$ στη μεταβλητή b .

 Στη C το ίσον (=) είναι ένας απλός τελεστής όπως και οι γνωστοί αριθμητικοί τελεστές (+, -, ...) και δεν αποτελεί μια εντολή ανάθεσης όπως σε άλλες γλώσσες προγραμματισμού.

Ο τελεστής ίσον (=) **καταχωρίζει** το αποτέλεσμα της παράστασης που βρίσκεται δεξιά του, στη **μεταβλητή** που εμφανίζεται στα αριστερά του. Αριστερά από έναν τελεστή ίσον (=) επιτρέπεται να εμφανίζεται **μόνο** το όνομα μιας μεταβλητής.

Μια παράσταση που περιέχει τον τελεστή ανάθεσης = έχει και αυτή μία τιμή, όπως έχει και οποιαδήποτε άλλη παράσταση. Η τιμή μιας τέτοιας παράστασης είναι η τιμή που καταχωρίζεται στη μεταβλητή αριστερά του ίσον. Τα παρακάτω παραδείγματα κάνουν σαφή τη χρήση του τελεστή =. Παραστάσεις που απαγορεύονται ρητά σε άλλες γλώσσες προγραμματισμού, όχι μόνο επιτρέπονται, αλλά χρησιμοποιούνται ευρέως από τους προγραμματιστές της C.

$$a = 5;$$

Ο τελεστής = καταχωρίζει το 5 στη μεταβλητή a και το αποτέλεσμα της παράστασης $a = 5$ είναι το 5.

$$b = a = 5;$$

Η παράσταση αυτή καταχωρίζει το 5 και στη μεταβλητή a , και στη μεταβλητή b . Αν σκεφτούμε αυτή την παράσταση ως $b = (a = 5)$, η παράσταση $a = 5$ καταχωρίζει στην a το 5 και έχει αποτέλεσμα 5. Έτσι, δεξιά από το πρώτο = η τιμή της παράστασης είναι 5, το οποίο καταχωρίζεται στην b . Το αποτέλεσμα όλης της παράστασης είναι πάλι το 5.

ΠΡΟΣΟΧΗ πρέπει να δοθεί στο γεγονός ότι μια μεταβλητή στη C, πριν να της ανατεθεί κάποια τιμή, έχει απροσδιόριστο περιεχόμενο (και όχι 0 ή κενό όπως συμβαίνει σε άλλες γλώσσες).

Για παράδειγμα, αν θεωρήσουμε τις παρακάτω προτάσεις:

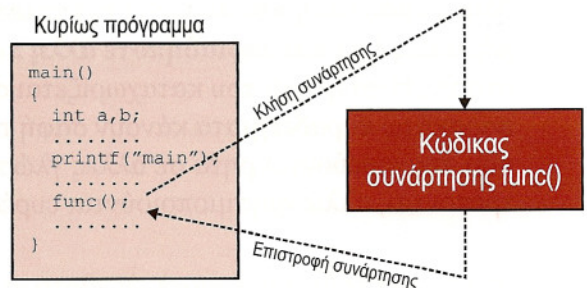
```
int a,b,c;  
a=5;  
c=a+b;
```




η μεταβλητή **c** θα πάρει απροσδιόριστη τιμή διότι στην **b** δεν έχει προηγουμένως ανατεθεί κάποια τιμή.

Πιο σύνθετα παραδείγματα θα αναφέρουμε σε επόμενες παραγράφους, όταν θα αναφερθούμε στους υπόλοιπους τελεστές της C και στην προτεραιότητα των πράξεων.

Συναρτήσεις: Μια συνάρτηση μπορεί από μόνη της να αποτελέσει μια πρόταση ή να μετέχει σε μια παράσταση.

Μια συνάρτηση καλείται χρησιμοποιώντας μόνο το όνομά της. Μόλις εκτελεστεί ο κώδικας της συνάρτησης, ο έλεγχος του προγράμματος επιστρέφει αμέσως μετά το σημείο κλήσης της συνάρτησης.



-  Κάθε συνάρτηση στη C επιστρέφει μία τιμή (εκτός αν έχει συγκεκριμένα δηλωθεί ότι δεν θα επιστρέψει τιμή).
-  Μια συνάρτηση στη C μπορεί να είναι μια συνάρτηση βιβλιοθήκης, δηλαδή μια έτοιμη συνάρτηση με προκαθορισμένη λειτουργία, ή να ορίζεται μέσα στο ίδιο το πρόγραμμα ως ξεχωριστό τμήμα προγράμματος.
-  Μια συνάρτηση παίρνει πληροφορίες από το πρόγραμμα που την καλεί (μέσω των παραμέτρων της), εκτελεί τη λειτουργία για την οποία σχεδιάστηκε, και τελικά επιστρέφει στο πρόγραμμα που την κάλεσε.



Μια συνάρτηση εκτελεί οπωσδήποτε μια συγκεκριμένη λειτουργία, μπορεί όμως να μην έχει καμία παράμετρο και να μην επιστρέφει καμία τιμή (ή η τιμή που επιστρέφει να μη χρησιμοποιείται από το πρόγραμμα).

Παραστάσεις με μέλη διαφορετικού τύπου

Σε μια παράσταση μπορούν να μετέχουν μέλη διαφορετικού τύπου. Το ερώτημα είναι τι τύπου θα είναι το αποτέλεσμα της παράστασης;

Αν ιεραρχήσουμε τους τύπους δεδομένων της C, η σειρά από τον ιεραρχικά χαμηλότερο προς τον υψηλότερο είναι: `char`, `int`, `float`, και `double`.

Το αποτέλεσμα μιας παράστασης είναι του ίδιου τύπου δεδομένων με τον τύπο του **ιεραρχικά υψηλότερου** μέλους της παράστασης.

```
int a,b;
float c;
a=10;
b=5;
c=(a+b)/2;
```

Στο παραπάνω παράδειγμα, ενώ περιμένουμε η τιμή του `c` να είναι 7.5 (από το $15/2$), θα είναι 7. Αυτό γίνεται διότι η παράσταση $(a+b)/2$ έχει αποτέλεσμα `int`, εφόσον όλα τα μέλη της είναι τύπου `int` και δεν επιστρέφει δεκαδικά ψηφία. Αν θέλουμε να έχουμε το σωστό αποτέλεσμα (το 7.5), θα πρέπει η παράσταση να γραφεί με τον επόμενο τρόπο:


```
c=(a+b)/2.0;
```

Σε αυτή την περίπτωση, επειδή η σταθερά 2.0 είναι τύπου `float`, η παράσταση $(a+b)/2.0$ επιστρέφει τιμή `float`. Στην παράσταση αυτή ο τύπος `float` είναι ο ιεραρχικά υψηλότερος από τους τύπους των μελών της παράστασης.

Λογικές παραστάσεις


Μια λογική παράσταση απαρτίζεται από μία ή περισσότερες **λογικές φράσεις** οι οποίες συνδέονται μεταξύ τους με συνδετικούς τελεστές. Μια λογική φράση, είναι μια σύγκριση με χρήση συγκριτικών τελεστών. Μια λογική φράση και, επομένως, μια λογική παράσταση, μπορεί να έχει την τιμή **Αλήθεια** ή την τιμή **Ψέμα**.

Για παράδειγμα, η παράσταση $a > 5$ είναι μία λογική φράση που έχει αποτέλεσμα **Αλήθεια** στην περίπτωση που η τιμή της μεταβλητής a είναι μεγαλύτερη από 5 και **Ψέμα** σε διαφορετική περίπτωση.

 Στη C, κάθε λογική παράσταση έχει αποτέλεσμα μία **αριθμητική** τιμή:
1 για αλήθεια και
0 για ψέμα

Στον επόμενο πίνακα αναφέρονται οι συγκριτικοί και συνδυαστικοί τελεστές που χρησιμοποιεί η C.

Συγκριτικοί τελεστές		Συνδυαστικοί τελεστές	
==	ίσο	&&	λογικό AND
!=	όχι ίσο (άνισο)		λογικό OR
>	μεγαλύτερο	!	λογικό NOT
>=	μεγαλύτερο ή ίσο		
<	μικρότερο		
<=	μικρότερο ή ίσο		

 Παρατηρούμε ότι ο **τελεστής της ισότητας είναι δύο ίσον (==)** και διαφέρει από τον αριθμητικό τελεστή ανάθεσης, το μονό ίσον (=). Προσοχή γιατί αυτή η λεπτή διαφορά είναι αιτία για αρκετούς πονοκεφάλους.

Οι συνδυαστικοί τελεστές && (AND) και || (OR) συνδέουν δύο λογικές φράσεις δημιουργώντας έτσι μία λογική παράσταση:

Λογική-φράση-A && Λογική-φράση-B

Λογική-φράση-A || Λογική-φράση-B

Το αποτέλεσμα των παραπάνω λογικών παραστάσεων φαίνεται στους επόμενους πίνακες αλήθειας των δύο τελεστών.

Πίνακας αλήθειας συνδυαστικού τελεστή && (AND)		
Λογική φράση A	Λογική φράση B	A && B
Αλήθεια	Αλήθεια	Αλήθεια
Αλήθεια	Ψέμα	Ψέμα
Ψέμα	Αλήθεια	Ψέμα
Ψέμα	Ψέμα	Ψέμα

Πίνακας αλήθειας συνδετικού τελεστή (OR)		
Λογική φράση A	Λογική φράση B	A B
Αλήθεια	Αλήθεια	Αλήθεια
Αλήθεια	Ψέμα	Αλήθεια
Ψέμα	Αλήθεια	Αλήθεια
Ψέμα	Ψέμα	Ψέμα


Ο τελεστής ! (NOT) δεν συνδέει λογικές φράσεις μεταξύ τους αλλά εφαρμόζεται σε μία μόνο λογική φράση (προς τα δεξιά του) και αντιστρέφει τη λογική της.

Πίνακας αλήθειας τελεστή ! (NOT)	
Λογική έκφραση A	! A
Αλήθεια	Ψέμα
Ψέμα	Αλήθεια

Όπως αναφέρθηκε και προηγουμένως, μια λογική παράσταση στη C επιστρέφει μία αριθμητική τιμή: 1 για αλήθεια και 0 για ψέμα. Επομένως παραστάσεις όπως:


```
a=3>2;
```

είναι απολύτως αποδεκτές. Στη συγκεκριμένη περίπτωση η μεταβλητή *a* θα πάρει την τιμή 1 διότι η λογική παράσταση *3>2*, ως αληθής, επιστρέφει τιμή 1. Θα διαπιστώσουμε αργότερα ότι αριθμητικές παραστάσεις και λογικές φράσεις μπορούν να συνυπάρχουν μέσα στην ίδια παράσταση.

 Όταν έχουμε πολλές συνδεδεμένες λογικές φράσεις, για να ορίσουμε την προτεραιότητα με την οποία θα υπολογιστούν χρησιμοποιούμε παρενθέσεις:

```
(a>b && a==c) || a<c
```

Πρώτα θα υπολογιστεί η λογική παράσταση μέσα στις παρενθέσεις.

 Στη περίπτωση που δεν υπάρχουν παρενθέσεις, η προτεραιότητα των τελεστών είναι καθορισμένη και αναφέρεται στο κεφάλαιο 4 (σελίδα 87).

Παραδείγματα κώδικα με παραστάσεις

Στα παραδείγματα που ακολουθούν θα χρησιμοποιήσουμε και δύο συναρτήσεις βιβλιοθήκης της C: την `printf()` και την `rand()`.

Δεδομένου ότι η C δεν έχει ενσωματωμένες εντολές για είσοδο και έξοδο πληροφοριών, τις λειτουργίες αυτές αναλαμβάνουν οι συναρτήσεις βιβλιοθήκης. Η

`printf()` χρησιμοποιείται για την εμφάνιση πληροφοριών στην οθόνη. Θα χρησιμοποιήσουμε την πιο απλή μορφή της συνάρτησης, η οποία απλώς εμφανίζει ένα σύνολο χαρακτήρων στην οθόνη:

`printf("Αυτό εμφανίζεται στην οθόνη")`

Όταν καλούμε την παραπάνω συνάρτηση, στην οθόνη εμφανίζεται το κείμενο μέσα στα εισαγωγικά. Ίσως φανεί παράξενο, αλλά η `printf()` επιστρέφει και μία τιμή, πέρα από τη λειτουργία που εκτελεί (την εμφάνιση των χαρακτήρων στην οθόνη). Η τιμή αυτή είναι ο αριθμός των χαρακτήρων που εμφάνισε, δηλαδή στη συγκεκριμένη περίπτωση ο αριθμός 27 (αλφαβητικοί χαρακτήρες μαζί με τα κενά διαστήματα).

Η συνάρτηση `rand()` δεν δέχεται παραμέτρους και επιστρέφει ως τιμή έναν τυχαίο ακέραιο αριθμό.

Το επόμενο πρόγραμμα υπολογίζει το μέσο όρο των περιεχομένων των μεταβλητών `a` και `b`.

```
main()
{
    int a,b;
    float c;
    a = rand();           // ◀ Στη μεταβλητή a καταχωρίζεται ο τυχαίος
                        // ακέραιος αριθμός που επιστρέφει η rand().
    b = 18;
    c = (a + b)/2.0;      // ◀ Στη μεταβλητή c καταχωρίζεται
                        // ο μέσος όρος των a και b
}
```

Οι δύο πρώτες προτάσεις είναι δηλωτικές και ενημερώνουν το μεταγλωττιστή για την ύπαρξη των τριών μεταβλητών `a`, `b` και `c`. Στις μεταβλητές `a` και `b` μπορούν να καταχωριστούν ακέραιοι αριθμοί, ενώ η `c` μπορεί να αποθηκεύσει ένα δεκαδικό αριθμό.

Σε άλλες γλώσσες προγραμματισμού, οι επόμενες τρεις εκτελέσιμες προτάσεις μπορεί να θεωρηθούν εντολές ανάθεσης για τις μεταβλητές `a`, `b` και `c` αντίστοιχα. Στη C, καμία από αυτές δεν θεωρείται εντολή (με την ακριβή έννοια του

όρου) αλλά είναι παραστάσεις που απλώς περιέχουν τον τελεστή ανάθεσης ίσον (=).

Αναφέρθηκε προηγουμένως ότι μία πρόταση σε ένα πρόγραμμα της C μπορεί να είναι μία δηλωτική πρόταση, μία εντολή, ή μία παράσταση. Είναι συντακτικά σωστό το πρόγραμμα που ακολουθεί:

```
main()
{
    int a,b;
    5+3;
    a = 6;
    8;
}
```

Και όμως, είναι !!!

Η δεύτερη πρόταση αποτελείται από την παράσταση 5+3, η οποία απλώς υπολογίζεται χωρίς το αποτέλεσμά της να χρησιμοποιείται πουθενά. Η πρόταση αυτή δεν επηρεάζει την λειτουργία του προγράμματος, είναι εντελώς άχρηστη, αλλά συντακτικά σωστή, όπως και η τελευταία (το "σκέτο" 8).

Δουλεύοντας με τη C πρέπει να ξεχάσουμε πολλούς από τους περιορισμούς άλλων γλωσσών και να μπορούμε στη δικιά της ελεύθερη και γοητευτική φιλοσοφία.

Η C και οι αγκύλες της

Μέχρι τώρα, περιοριστήκαμε στη χρήση των αγκυλών {} για την ένδειξη της αρχής (με την αριστερή { αγκύλη) και του τέλους (με την δεξιά } αγκύλη) του συνόλου των προτάσεων μιας συνάρτησης:

```
main()
{
    .....
    .....
    .....
}
```

← Αρχή προτάσεων

← Τέλος προτάσεων

Οι αγκύλες όμως δεν έχουν μόνο αυτή τη χρήση. Αντίθετα, αποτελούν ένα βασικό κομμάτι της φιλοσοφίας της C για τη δημιουργία δομημένων και "ευανάγνωστων" προγραμμάτων.

Σύνθετη πρόταση (compound statement)


Σύνθετη πρόταση είναι μια ομάδα από απλές προτάσεις που περιέχονται μέσα σε αγκύλες:

```
{
    a=4;
    b=8;
    printf("Αυτή είναι μία σύνθετη πρόταση");
}
```

 Μια σύνθετη πρόταση μπορεί να περιέχει οποιονδήποτε αριθμό απλών προτάσεων, αλλά και άλλες σύνθετες προτάσεις.

```
{
    a=4;
    {
        m=8;
        printf("Αυτή είναι μία ακόμη σύνθετη πρόταση");
    }
    printf("Αυτή είναι μία σύνθετη πρόταση");
}
```

 Οι απλές προτάσεις τελειώνουν πάντα με ελληνικό ερωτηματικό (;) ενώ οι σύνθετες όχι.

 Ο μεταγλωττιστής της C μεταχειρίζεται μια σύνθετη πρόταση όπως οποιαδήποτε απλή πρόταση. Έτσι, όταν με τον όρο "πρόταση" εννοείται είτε απλή είτε σύνθετη πρόταση.

Ο προ-μεταγλωττιστής της C

Μέσα στον πηγαίο κώδικα του προγράμματός μας είναι δυνατόν να περιλάβουμε οδηγίες προς το μεταγλωττιστή της C. Τέτοιες οδηγίες που θα συναντήσουμε συνήθως σε ένα πρόγραμμα της C είναι οι οδηγίες **#include** και **#define**.

Η οδηγία `#include`


Η οδηγία `#include` αναγκάζει το μεταγλωττιστή της C να συμπεριλάβει κατά τη διαδικασία της μεταγλώττισης και κάποιο άλλο πηγαίο αρχείο, στο οποίο συνήθως δηλώνονται συναρτήσεις βιβλιοθήκης.

Για παράδειγμα, το αποτέλεσμα της πρότασης

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

είναι να συμπεριλάβει ο μεταγλωττιστής κατά τη διαδικασία της μεταγλώττισης και το αρχείο `stdio.h`, στο οποίο δηλώνονται οι περισσότερες συναρτήσεις εισόδου και εξόδου της C.

Η χρήση της `#include` αναλύεται αργότερα, στο Κεφάλαιο 9.

 Προς το παρόν αρκεί να γνωρίζουμε ότι για να χρησιμοποιήσουμε μια συνάρτηση βιβλιοθήκης, θα πρέπει να χρησιμοποιήσουμε την `#include` για να συμπεριλάβουμε το αρχείο στο οποίο δηλώνεται.

 ΠΡΟΣΟΧΗ η οδηγία `#include` δεν τερματίζεται με ερωτηματικό (;).

Η οδηγία `#define`

Η οδηγία `#define` ορίζει ένα αναγνωριστικό και ένα σύνολο χαρακτήρων που θα αντικαταστήσει αυτό το αναγνωριστικό κατά τη διαδικασία της μεταγλώττισης του πηγαίου κώδικα. Η σύνταξη της οδηγίας είναι:

`#define` αναγνωριστικό χαρακτήρες

όπου το **αναγνωριστικό** αντικαθίσταται στο στάδιο της μεταγλώττισης από τους **χαρακτήρες**, σε όποιο σημείο του πηγαίου κώδικα και αν βρεθεί.

Συνήθως χρησιμοποιούμε την οδηγία `#define` για να ορίσουμε κάποιες σταθερές του προγράμματός μας:

```
#define PI 3.141593
main()
{
    float aktina, emvadon;
    aktina=10;
```



```
embadon=aktina*aktina*PI;
```

```
}
```


Το παραπάνω πρόγραμμα υπολογίζει το εμβαδόν ενός κύκλου ακτίνας 10 μέτρων. Ο προ-μεταγλωττιστής της C αντικαθιστά κατά τη διαδικασία της μεταγλώττισης το `PI` με την τιμή 3.141593.

Με τις παρακάτω οδηγίες:

```
#define ONE 1
```

```
#define TWO 2
```

μια πρόταση `C=ONE+TWO` θα είναι σωστή και θα καταχωρίσει την τιμή 3 στη μεταβλητή `C`.

 Πρέπει να γίνει κατανοητό ότι στα προηγούμενα παραδείγματα τα `PI`, `ONE`, και `TWO` **δεν** είναι μεταβλητές.

 ΠΡΟΣΟΧΗ: Η οδηγία `#define` δεν τερματίζεται με ερωτηματικό (;).

Παραδείγματα

Π.1 Στο παρακάτω πρόγραμμα υπάρχουν δύο συνήθη συντακτικά λάθη:

α) Οι δηλωτικές προτάσεις πρέπει να είναι οι πρώτες προτάσεις στον κορμό μιας συνάρτησης. Συγκεκριμένα, η πρόταση `float c` εμφανίζεται αφού έχουν μεσολαβήσει 2 εκτελέσιμες προτάσεις.

β) Η τελευταία πρόταση δεν τερματίζεται με ερωτηματικό (;).

```
main()
{
    int a,b;
    a = 10;
    b = 18;
    float c;
    c = (a + b) / 2
}
```

Π.2 Στο παρακάτω πρόγραμμα δηλώνεται η μεταβλητή **a** (η οποία δεν χρησιμοποιείται αργότερα) και εμφανίζεται στην οθόνη η φράση **C is the best**. Η `printf()`, πέρα από την εμφάνιση των χαρακτήρων, επιστρέφει και μία τιμή (το πλήθος των χαρακτήρων που εμφάνισε) η οποία στη συγκεκριμένη περίπτωση χάνεται.

```
main()
{
    int a;
    printf("C is the best");
}
```

Στον κώδικα που ακολουθεί, η τιμή που επιστρέφει η `printf()` ανατίθεται στη μεταβλητή **a**, η οποία έχει μετά την πρόταση `a=printf("C is the best")` την τιμή 13 (το πλήθος των αλφαβητικών χαρακτήρων μαζί με τα κενά διαστήματα).

```
main()
{
    int a;
    a = printf("C is the best");
}
```

Π.3 Τι τιμή θα πάρει η μεταβλητή **b** στο παρακάτω πρόγραμμα;

```
main()
{
    int a,b;
    a = rand();
    rand();
    b = 4 + a = 5 + printf("telos");
}
```

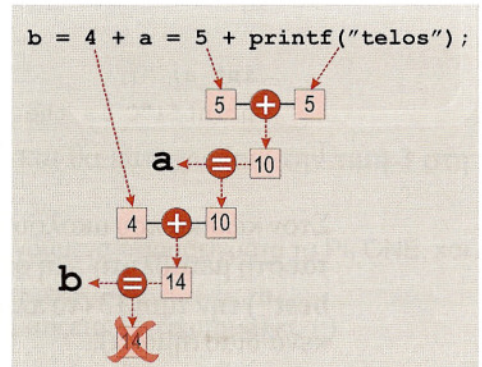
Η παράσταση `a = rand()` έχει αποτέλεσμα την καταχώριση ενός τυχαίου αριθμού στη μεταβλητή **a**.

Η πρόταση **rand()**; καλεί τη συνάρτηση **rand()** που επιστρέφει έναν τυχαίο αριθμό, ο οποίος όμως χάνεται. Ουσιαστικά αυτή η πρόταση είναι άχρηστη.

Η τελευταία πρόταση αποτελείται από μια παράσταση, όχι και τόσο παράλογη αν αναλογιστούμε τη διαφορετική φιλοσοφία της C.

Η συνάρτηση **printf("telos")** επιστρέφει την τιμή 5.

- Το 10 (από 5+5) καταχωρίζεται στην **a** και η όλη παράσταση επιστρέφει τιμή 10.
- Το 14 (από 10+4) καταχωρίζεται στη μεταβλητή **b**.
- Η όλη παράσταση επιστρέφει τιμή 14, η οποία δεν χρησιμοποιείται.



Η σειρά εκτέλεσης των πράξεων θα γίνει ξεκάθαρη όταν θα αναφερθούμε στην προτεραιότητα των τελεστών στο κεφάλαιο 4 (σελίδα 87).

Ανασκόπηση Κεφαλαίου 2

- Κάθε πρόγραμμα της C περιέχει υποχρεωτικά μία συνάρτηση **main()** η οποία είναι εκείνη που εκτελείται πρώτη. Κάθε πρόγραμμα της C μπορεί να περιέχει και άλλες συναρτήσεις, εκτός από τη **main()**, οι οποίες για να εκτελεστούν πρέπει να κληθούν.
- Κάθε μεταβλητή που χρησιμοποιείται πρέπει να δηλώνεται. Οι δηλωτικές προτάσεις των μεταβλητών τοποθετούνται στην αρχή μιας συνάρτησης και πριν από τις εκτελέσιμες προτάσεις.
- Οι βασικοί τύποι μεταβλητών στη C είναι ο τύπος **char**, ο τύπος **int**, και ο τύπος **float**.
- Οι αριθμητικές παραστάσεις είναι πράξεις μεταξύ αριθμητικών δεδομένων και έχουν αποτέλεσμα έναν αριθμό.

- Οι λογικές παραστάσεις χρησιμοποιούν συγκριτικούς και συνδυαστικούς τελεστές και έχουν αποτέλεσμα **αλήθεια** (τιμή 1) ή **ψέμα** (τιμή 0).
- Ο τελεστής `=` καταχωρίζει το αποτέλεσμα της παράστασης στα δεξιά του στη μεταβλητή που βρίσκεται αριστερά του, π.χ. `a=b+3`
- Μια σύνθετη πρόταση αποτελείται από απλές προτάσεις μέσα σε αριστερή και δεξιά αγκύλη.

Ασκήσεις Κεφαλαίου 2

- 2.1** Τι θα περιέχουν οι μεταβλητές `a`, `b`, και `c` μετά το τέλος του παρακάτω κώδικα: ★

```
main()
{
    int a,b,c=3;
    a=b=2;
    a=c+b;
}
```

- 2.2** Τι θα περιέχουν οι μεταβλητές `a`, `b`, και `c` μετά το τέλος του παρακάτω κώδικα: ★

```
#define MM 23
main()
{
    const int c=3;
    int a,b;
    a=4+b=2;
    b=c+b+MM;
}
```

- 2.3** Εντοπίστε τα λάθη στον παρακάτω κώδικα: ★★

```
#define MM 23;
```

```
main()
{
    const int c=3;
    int a,b;
    a=2;
    float d;
    d=4.3
    a=4+b=2;
    MM=10;
    3=a;
    c=c+b+MM;
}
```

- 2.4** Τι θα περιέχουν οι μεταβλητές **a**, **b**, και **c** μετά το τέλος του παρακάτω κώδικα: ★★

```
main()
{
    int a,b,c=3;
    a=b=2;
    a=c>b;
    b=b==1;
    c=printf("τέλος");
}
```

- 2.5** Ποια από τα παρακάτω αληθεύουν: ★

- ☐ Δηλωτικές προτάσεις μπορούν να μπουν σε οποιοδήποτε σημείο του προγράμματος.
- ☐ Ένα πρόγραμμα της C μπορεί να περιέχει πολλά υποπρογράμματα (συναρτήσεις).
- ☐ Μια λογική παράσταση έχει τιμή 1 ή 0.
- ☐ Μια μεταβλητή στη C, πριν της δοθεί τιμή, έχει τιμή 0.
- ☐ Η οδηγία `#define` χρησιμοποιείται για να ορίσει μία σταθερά του προγράμματος μας.

- 2.6** Με δεδομένες τις τιμές των μεταβλητών a, b , και c σε 5, 10, και 15 αντίστοιχα, σημειώστε την τιμή (1 για αλήθεια, 0 για ψέμα) των παρακάτω λογικών παραστάσεων: ★ ★

Λογική παράσταση	Τιμή
$a == (c - b)$	
$a > b \ \ b > c$	
$a == 5 \ \&\& \ c = 15$	
$a == 5 \ \&\& \ c > 20$	

- 2.7** Να γραφεί πρόγραμμα το οποίο να αποθηκεύει τους αριθμούς 3, 7, και 21 σε τρεις θέσεις μνήμης. Κατόπιν, να υπολογίζει και να αποθηκεύει σε μία τέταρτη θέση μνήμης το μέσο όρο τους. ★ ★
- 2.8** Να τροποποιηθεί το προηγούμενο πρόγραμμα ώστε να υπολογίζει το μέσο όρο τριών τυχαίων αριθμών. ★ ★