Εισαγωγή στην Πληροφορική & στον Προγραμματισμό

Αρχές Προγραμματισμού Η/Υ (με τη γλώσσα C)

Διάλεξη #2

29 Μαρτίου 2023

Παναγιώτης Παύλου

c-programming-24@allos.gr

Ορισμός Συναρτήσεων

Πως δημιουργείται μια συνάρτηση

Ορισμός συνάρτησης

Για να δημιουργηθεί μία νέα συνάρτηση σε ένα πρόγραμμα θα πρέπει να δοθεί σε αυτό ο **ορισμός** (**definition**) της. Για παράδειγμα ο ορισμός μιας μαθηματικής συνάρτησης π.χ.

$$f(x) = x^2$$

η οποία θα διαχειρίζεται πραγματικούς αριθμούς γράφεται:

```
function squareOf(x)
{
   return x*x;
}
```

Ο **ορισμός** αυτός περιλαμβάνει με τη σειρά:

1. Την **λέξη κλειδί της C*** `function` που δείχνει ότι ακολουθεί **ορισμός συνάρτησης**
2. Το **όνομα** της συνάρτησης, το οποίο ακολουθεί τους κανόνες των identifiers, άρα πρέπει να είναι μοναδικό σε όλο το πρόγραμμα
3. Οι **παράμετροι** (parameters) της συνάρτησης, δηλαδή τις τιμές που δέχεται ως δεδομένα, μέσα σε παρενθέσεις, χωρισμένες με κόμμα μεταξύ τους
4. Το **σώμα των εντολών** της συνάρτησης, δηλωμένο ως block (δηλαδή οι εντολές της συνάρτησης μέσα σε άγκιστρα)

Σημεία προσοχής

Παράμετροι

Στον ορισμό μίας συνάρτησης οι παράμετροι λειτουργούν ως τοπικές μεταβλητές, ακριβώς όπως και αν ήταν δηλωμένες στην αρχή του σώματος της συνάρτησης.

Προσέξτε τη διαφορά από την έννοια των ορισμάτων που αναφέραμε νωρίτερα : Κάθε φορά που καλείται η συνάρτηση, οι τιμές που παίρνουν οι παράμετροι είναι αντίγραφα των αντίστοιχων ορισμάτων που δίνονται.

Επιστρεφόμενη τιμή

Κάθε συνάρτηση πρέπει να έχει τουλάχιστον μία εντολή **`return`**. Η εντολή αυτή όταν εκτελεστεί τερματίζει την εκτέλεση της συνάρτησης και αμέσως μετά τη λέξη `return` δίνεται η τιμή που επιστρέφει η συνάρτηση (δηλαδή το αποτέλεσμά της). Εάν η συνάρτηση δεν επιστρέφει κάποια τιμή, τότε η **`return`** δεν θα έχει δίπλα της κάποια τιμή. Σε κάθε περίπτωση, όπως όλες οι εντολές, τελειώνει με τον τελεστή `;`

Όταν η τελευταία εντολή μιας συνάρτησης είναι η **`return`** και η συνάρτηση δεν επιστρέφει τιμή, μόνο τότε η **`return`** μπορεί να παραληφθεί. Κατά την περίοδο εκμάθησης όμως καλύτερα να γράφεται πάντα.

Τοποθέτηση του ορισμού

Ο ορισμός της συνάρτησης πρέπει να γίνεται πάντα πριν την χρήση της (όπως και οι δηλώσεις των μεταβλητών).

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Σχόλια + Μορφοποίηση = Λιγότερα λάθη

Το ζήτημα της μείωσης των σφαλμάτων εξαρχής επιτρέπει τη μείωση του χρόνου του debugging. Σε αυτό συμβάλλουν πολύ:

Τα **σχόλια** στον κώδικα, δηλαδή κείμενα που δεν εκτελούνται ή χρησιμοποιούνται, αλλά αποτελούν «σημειώσεις» του προγραμματιστή για να μπορεί να επανέλθει ο ίδιος στον κώδικά του μετά από καιρό ή να συνεργαστεί με άλλον χωρίς να εξηγεί τα πάντα.

Τα ελάχιστα σχόλια είναι κυρίως οδηγίες χρήσης για τις συναρτήσεις, αλλά και επεξηγήσεις του ρόλου της κάθε μεταβλητής.

Η σωστή **μορφοποίηση** επιτρέπει την εύκολη αναγνώριση των block εντολών, κάνει ευανάγνωστες τις παραστάσεις και γενικά βοηθά στην γρηγορότερη κατανόηση του κώδικα που εμφανίζεται στην οθόνη.

Σχόλια – Comments

Τα σχόλια είναι δύο τύπων:

- Τα σχόλια **γραμμής** ή line comments
  που ξεκινούν με // και εκτείνονται μέχρι το τέλος της γραμμής. Συνήθως χρησιμοποιούνται για να σχολιάσουν ένα συγκεκριμένο σημείο του προγράμματος
- Τα σχόλια **περιοχής** ή block comments
  που ξεκινούν με /* και εκτείνονται μέχρι το */ Συνήθως χρησιμοποιούνται για να σχολιάσουν μία συνάρτηση ή για να «απενεργοποιήσουν» μία περιοχή του κώδικα. Θέλουν προσοχή καθώς εάν σχολιαστεί μία περιοχή του κώδικα που περιέχει ήδη ένα τέτοιο σχόλιο μέσα της, τότε η περιοχή τερματίζεται στο πρώτο */ που θα συναντηθεί.

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Εμβάθυνση στις συναρτήσεις

Ο μηχανισμός της κλήσης και οι μεταβλητές τους

Εμβέλεια τοπικών μεταβλητών

Τοπική μεταβλητή, όπως ήδη αναφέρθηκε, είναι η κάθε μεταβλητή που δηλώνεται στα πλαίσια ενός block εντολών. Η μεταβλητή δημιουργείται στη μνήμη με τη δήλωσή της και συνεχίζει να υπάρχει μέχρι η ροή εκτέλεσης να βγει από το συγκεκριμένο block. Η έκταση στην οποία μπορεί να χρησιμοποιηθεί αυτή η μεταβλητή ονομάζεται **εμβέλεια της μεταβλητής**. Στις τοπικές μεταβλητές ταυτίζεται με την έκταση του block στο οποίο ορίζονται

Μέσα στο ίδιο block εντολών δεν επιτρέπεται να δηλωθεί δεύτερη μεταβλητή με το ίδιο όνομα.

```
function main() {
    let A = 1;

    smPrint("Program starting...\n");
    { // Μπορώ να έχω ένα block χωρίς λόγο!
        smPrint("Inside the block\n");
        let x;
        x = 10;
        x = x + A;
        smPrint("Block completing...\n");
    }

    smPrint("Out of block!\n");
    return 0;
}
```

Επισκίαση μεταβλητών

Κάθε **τοπική μεταβλητή** πρέπει να έχει μοναδικό όνομα στο block της, όμως σε άλλα block – ακόμα και ένθετα/εμφωλευμένα (nested) – δεν υπάρχει περιορισμός. Έτσι ο δίπλα κώδικας είναι απόλυτα ορθός.

Η μεταβλητή Z στο εξωτερικό block είναι διαφορετική από την Z στο εσωτερικό. Για όσο υπάρχει η Z του εσωτερικού block ο κώδικας δεν έχει τρόπο να αναφερθεί στην Z του εξωτερικού block. Αυτό ονομάζεται επισκίαση (shadowing) της μεταβλητής Z.

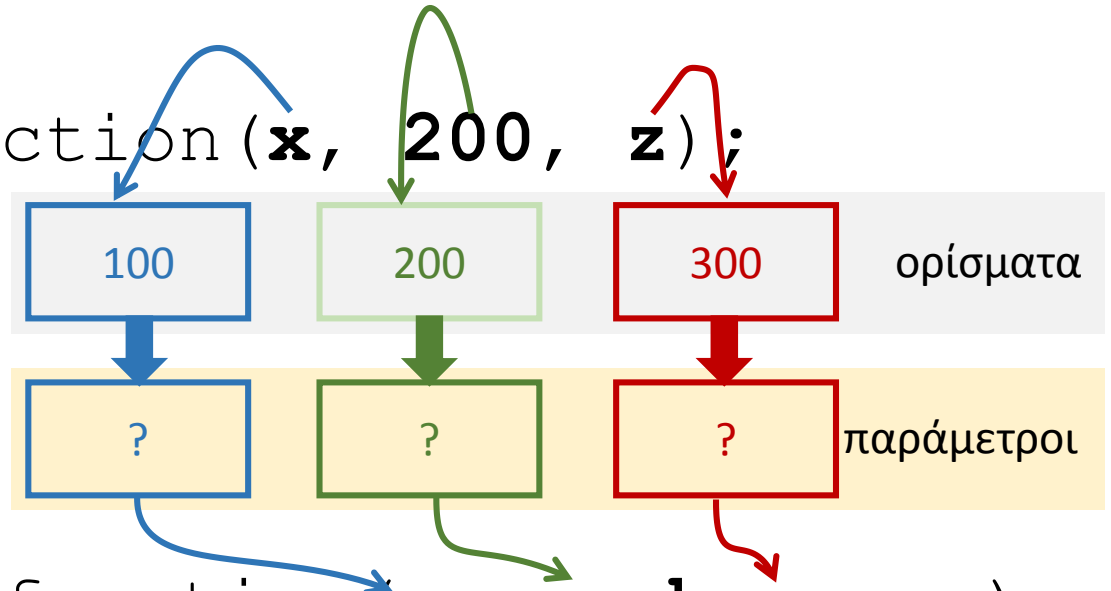
```
function main() {
    let Z = 0;
    smPrint("Program starting... %\n",Z);
    {
        let Z = 10;
        smPrint("After definition: %\n",Z);
        Z = Z + 1;
        smPrint("Block end: %\n",Z);
    }

    smPrint("Out of block! %\n", Z);
    return 0;
}
```

Call by value – Κλήση με τιμή

Μία κλήση συνάρτησης:

```
some_function(x, 200, z);
```



Και ο ορισμός της:

```
function some_function(  a,   b,   z   )
```

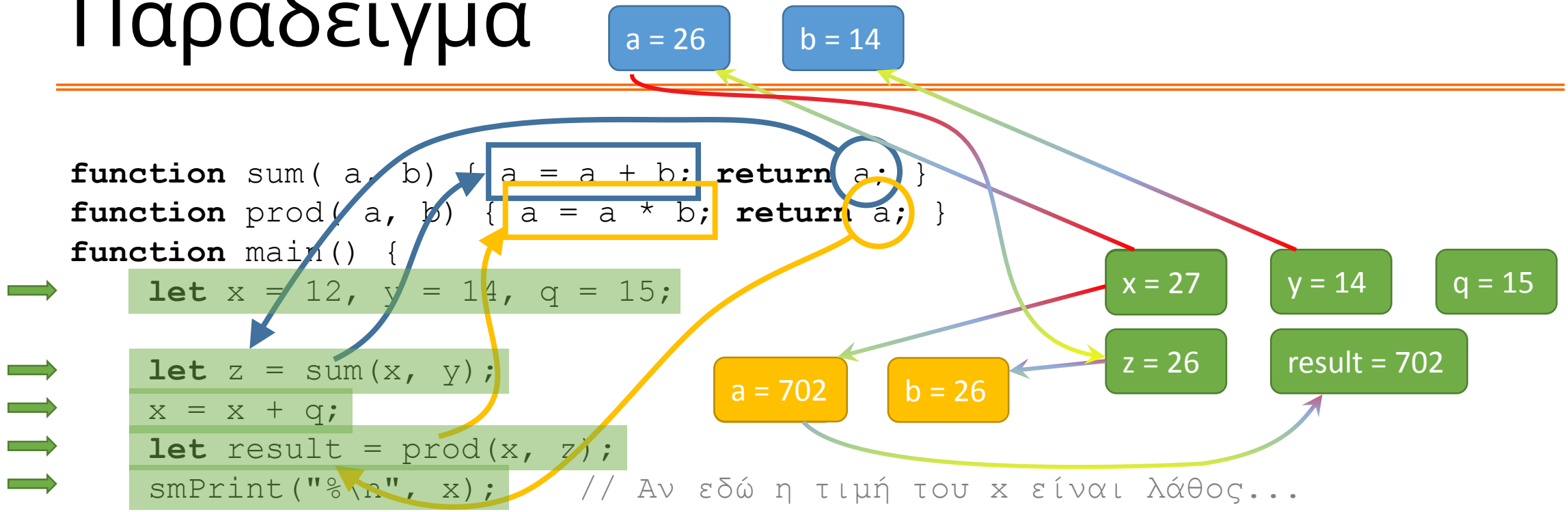
Για να γίνει η κλήση οι τιμές των **ορισμάτων** x,y,z **αντιγράφονται** στις **παραμέτρους** a,b,z και προσέξτε ότι οι a,b,z είναι άλλες μεταβλητές από τις x,y,z !

Συνέπειες του Call-by-value

Η κλήση με τιμή (call-by-value) είναι ο μοναδικός τρόπος που υποστηρίζει η C για την κλήση συναρτήσεων. Αυτό έχει κάποιες συνέπειες:

- Αφού δημιουργούνται αντίγραφα των τιμών, οι όποιες αλλαγές στις παραμέτρους μέσα στο σώμα της συνάρτησης δεν επιδρούν στις τιμές των ορισμάτων. Οι παράμετροι δρουν ως τοπικές μεταβλητές μέσα στη συνάρτηση.
- Καμία κλήση συνάρτησης δεν μπορεί να αλλάξει τις τιμές των ορισμάτων που χρησιμοποιούνται σε αυτή. Αυτό μειώνει τον χρόνο του debugging αισθητά!

Παράδειγμα



Αν η τιμή του `x` βρεθεί να είναι λανθασμένη στη γραμμή με το σχετικό σχόλιο, τότε δεν μπορεί να οφείλεται στη συνάρτηση `sum`.

Ο προγραμματιστής χρειάζεται να κοιτάξει μόνο τις γραμμές όπου γράφεται `x=` άρα κερδίζει πολύ χρόνο κατά το debugging!

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

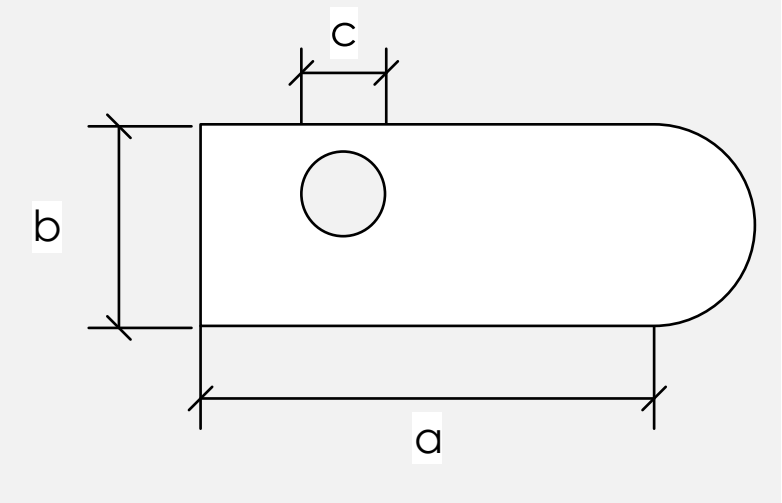
  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Εφαρμογή #1

Να γραφτεί ένα πρόγραμμα το οποίο να:

1. υπολογίζει το εμβαδό του σχήματος
2. ύστερα να το στρογγυλοποιεί προς τα πάνω στο αμέσως μεγαλύτερο πολλαπλάσιο του 100
3. κατόπιν να υπολογίζει υπεραπλουστευτικά πόσα κουτάκια μπογιάς που το κάθε ένα καλύπτει συγκεκριμένη επιφάνεια θα χρειάζονταν για να καλυφθεί η επιφάνεια του σχήματος
4. τέλος να υπολογίζει πόσο μέρος του «τελευταίου κουτιού» περισσεύει και να το παρουσιάζει ως αριθμό και ως ποσοστό του περιεχομένου του.




```
function main() {
    let a = 104, b = 401, c = exp(1);
    let Ebox = a * b;
    let Er = M_PI * pow( b/2.0, 2.0 ) / 2.0;
    let Ecrc = M_PI * pow( c/2.0, 2.0 );
    let A = Ebox + Er - Ecrc;
    let factor = pow( 10, 2 );
    let Ar = ceil( A / factor ) * factor;
    smPrint("% -> %\n", A, Ar);
    let bucketSize = 123;
    let buckets = floor(A/ bucketSize)+1;
    smPrint(" Buckets : %\n", buckets);
    let remainder = buckets * bucketSize - A;
    smPrint("Loss: % (% %%)\n",
            remainder,
            100 * remainder / bucketSize);
    return 0;
}
```

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Λογικές Παραστάσεις

Προτάσεις που μπορούν να χαρακτηριστούν αληθείς ή ψευδείς

Λογικές ποσότητες

Μία λογική ή boolean ποσότητα είναι η απλούστερη πληροφορία που μπορεί να παρασταθεί και ταιριάζει και με τη δυαδική λογική του Η/Υ. Η λογική ποσότητα έχει δύο τιμές **αληθή (true)** ή **ψευδή (false)** όπως και ένα δυαδικό ψηφίο.

Μάλιστα ως αριθμητικές ποσότητες αντιστοιχούν: το αληθές στον αριθμό **1** και το ψευδές στο **0** .

Γι'αυτό οι όροι **αληθές**, **true** και **1** θα χρησιμοποιούνται εναλλάξιμα και ομοίως οι όροι **ψευδές**, **false** και **0**.

Στην πράξη όμως ο υπολογιστής θεωρεί το `0` ψευδές και κάθε τι μη μηδενικό αληθές. Έτσι π.χ. και το `-4` αλλά και το `12.34e1` θεωρούνται αληθείς ποσότητες αφού δεν είναι μηδενικές!

```
function main() {
        let isRed = true;
        let isBad = false;
        return 0;
}
```

Τελεστές σύγκρισης

Οι κατεξοχήν παραστάσεις που δίνουν λογικό αποτέλεσμα είναι οι συγκρίσεις μεταξύ τιμών. Οι συγκρίσεις γίνονται μεταξύ αριθμητικών τιμών με τους ίδιους κανόνες όπως και οι αριθμητικές πράξεις.

Τελεστής	Αντίστοιχο λεκτικό
<	Μικρότερο
<=	Μικρότερο ή ίσο
>	Μεγαλύτερο
>=	Μεγαλύτερο ή ίσο
==	Ίσο
!=	Διάφορο

Δηλαδή συγκρίνουν αριθμητικές ποσότητες μεταξύ τους, αφού πρώτα τις μετατρέψουν στον ίδιο τύπο δεδομένων.

Π.χ.

```
a < 100
c >= d
1000 == x
let Q = a != b;
```

Τελεστές bool (1/2)

Άλγεβρα bool ονομάζεται η άλγεβρα μεταξύ λογικών ποσοτήτων. Οι βασικές πράξεις της που υποστηρίζονται και σε επίπεδο επεξεργαστή και παράγουν λογικές ποσότητες είναι:

- η άρνηση (NOT) που αντιστοιχεί στον μοναδιαίο τελεστή **!**
  Το αποτέλεσμα της άρνησης είναι η αντίθετη λογική τιμή
- η σύζευξη (AND) που αντιστοιχεί στον τελεστή **&&**
  Το αποτέλεσμα του AND είναι αληθές μόνο όταν και οι δύο τελεσταίοι (**a** και **b**) είναι true, όπως ο πολλαπλασιασμός μεταξύ 0 και 1.
- η διάζευξη (OR) που αντιστοιχεί στον τελεστή **||**
  Το αποτέλεσμα του OR είναι ψευδές μόνο όταν και οι δύο τελεσταίοι (**a** και **b**) είναι false, όπως η πρόσθεση μεταξύ 0 και 1.

x	!x
0	1
1	0

a	b	a && b
0	0	0
1	0	0
0	**1**	0
1	**1**	**1**

a	b	a || b
0	0	0
1	0	**1**
0	**1**	**1**
1	**1**	**1**

Τελεστές bool (2/2)

Τέλος πρέπει να δοθεί προσοχή στα εξής δύο σημεία.

Το ένα είναι η προτεραιότητά των τελεστών, η οποία ταυτίζεται με τη σειρά της προηγούμενης λίστας (NOT , AND , OR).

Επίσης σε σχέση με τους τελεστές σύγκρισης οι AND και OR έχουν χαμηλότερη προτεραιότητα, ενώ ο NOT υψηλότερη.

Προτεραιότητα
( )
!
&&
||

Δείτε τον πίνακα προτεραιότητας στις σημειώσεις (σελ. 10 στην έκδοση 2020.1)

Το άλλο αφορά τον τρόπο υπολογισμού των λογικών παραστάσεων. Στα OR και AND, εφόσον από τον υπολογισμό του 1ου τελεσταίου προκύπτει το αποτέλεσμα της παράστασης (δηλαδή στο OR αν είναι true και στο AND αν είναι false), τότε δεν υπολογίζεται το 2ο μέρος της παράστασης (ο 2ος τελεσταίος), έτσι:

```
x < 5 || test(x) == a
```

στο παραπάνω δεν εκτελείται ποτέ η συνάρτηση `test(x)` για `x<5`

Προτεραιότητα τελεστών

Μέχρι τώρα η προτεραιότητα των τελεστών που έχουμε μάθει είναι:

Τελεστής	Περιγραφή
( )	Παρενθέσεις και κλήση συνάρτησης
+ - !	Πρόσημο και άρνηση
* / %	Πολλαπλασιασμός , Διαίρεση , Υπόλοιπο
+ -	Πρόσθεση και αφαίρεση
< <= > >=	Ανισότητες
== !=	Ίσο & Διάφορο
&&	Σύζευξη
||	Διάζευξη
=	Ανάθεση τιμής

Παραδείγματα

Οι παρακάτω παραστάσεις είναι αληθείς όταν...

... η μεταβλητή `x` βρίσκεται στο διάστημα [`a b`)

```
a <= x && x < b
```

... μόνο ένα από τα `p` και `q` είναι αληθές

```
(p && !q) || (!p && q)
```

... το `x` δεν συμπίπτει με τα `A` και `B`

```
x!=A && x!=B
```

... οι `x, y, z` είναι διάφορες μεταξύ τους

```
x!=y && y!=z && z!=x
```

Τυπικά λάθη

Οι παρακάτω παραστάσεις είναι λανθασμένες επειδή...

... το = είναι ο τελεστής ανάθεσης και όχι σύγκρισης, αυτό σαν λογική παράσταση δίνει πάντα αληθές (αφού το 5 είναι μη μηδενικό)

```
x = 5
```

... οι συγκρίσεις δεν μπορεί να είναι πολλαπλές

```
A < x < B
```

... τα `|` , `^` και `&` δεν είναι λογικοί τελεστές αλλά αφορούν bits γι'αυτό δεν τα χρησιμοποιούμε για κανένα λόγο σε λογικές παραστάσεις. Αν x = 5 και y = 10 το παρακάτω επιστρέφει false παρότι είναι μη μηδενικά ενώ το `x && y` επιστρέφει true

```
x & y
```

... χρειάζεται πολύ προσοχή στην προτεραιότητα. Π.χ. Είναι τα x και y εκτός ορίων;

Εδώ θα εκτελεστεί πρώτα άρα θα χρειαστούν παρενθέσεις

```
( x < 0 || x > A )&&( y < 0 || y > A )
```

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Εφαρμογή #2

Οι συναρτήσεις που επιστρέφουν λογικές τιμές συνήθως ονομάζονται με όνομα που ξεκινά το is ώστε να "ρωτά" για το αν υπάρχει ή αν ισχύει κάτι.

Ας γραφεί μία τέτοια συνάρτηση που να επιστρέφει εάν ένας ακέραιος είναι ζυγός.

```
function isEven(N)
```

Εφαρμογή #3

Ζητάμε έναν κώδικα που να επιστρέφει το υπ'αριθμόν **d** ψηφίο ενός δεκαψήφιου αριθμού **N**. Τον αριθμό τον θεωρούμε δεκαψήφιο και ας μην είναι. Η αρίθμιση των ψηφίων ξεκινά από το 1 μέχρι το 10 και είναι από αριστερά προς τα δεξιά. Δηλαδή το 10ο ψηφίο είναι αυτό των μονάδων.

```
function digitOf10 (N, d)
```

Για παράδειγμα:

```
digitOf10( 0015008345, 9 ) → 4
digitOf10( 0015008345, 4 ) → 5
digitOf10( 0015008345, 7 ) → 8
```

Εφαρμογή #3 (σκεπτικό)

9 8 7 6 5 4 3 2 1 0

10^9 10^8 10^7 10^6 10^5 10^4 10^3 10^2 10^1 10^0

1 2 3 4 5 6 7 8 9 10

0 0 1 5 0 0 8 **3** 4 5

3 4 5

4 5