Εισαγωγή στην Πληροφορική & στον Προγραμματισμό

2

Αρχές Προγραμματισμού Η/Υ (με τη γλώσσα C)

Διάλεξη #2

Παναγιώτης Παύλου

intro-hy-24@allos.gr

Ορισμός Συναρτήσεων

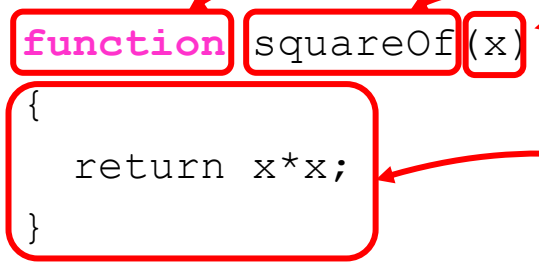
Πως δημιουργείται μια συνάρτηση

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:



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

  intro-hy-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, ούτε χρησιμοποιούνται από τον υπολογιστή, 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.

Μορφοποίηση

Η σωστή μορφοποίηση του κώδικα βοηθά στη εποπτική κατανόησή της δομής του και στην πιο άμεση και σαφή αναγνώριση των εντολών. Γίνεται αποκλειστικά και μόνο για τον προγραμματιστή που θα δει τον κώδικα για να καταλάβει τι συμβαίνει, είτε λόγω του ότι έχει να ασχοληθεί μαζί του πολύ καιρό, είτε επειδή τον έχει γράψει άλλος.

Η μορφοποίηση του κώδικα δεν λαμβάνεται υπόψη από τον υπολογιστή, γι' αυτό και πρέπει να ταιριάζει με τους τελεστές που χρησιμοποιούνται, πχ οι εσοχές να ταιριάζουμε με τα άγκιστρα.

Μερικές καλές πρακτικές μορφοποίησης (και όχι μόνο) είναι οι ακόλουθες:

1. Χρήση εσοχών: Όταν ανοίγουμε άγκιστρα, ο κώδικας που περιέχουν να είναι ένα tab (2-4 κενά) δεξιότερα του κώδικα που προηγείται.
2. Τοποθέτηση αγκίστρων: Το άνοιγμα ενός άγκιστρου να είναι στην ίδια γραμμή με την εντολή που αφορά. Πχ `function a(x) {`
3. Πλήθος χαρακτήρων: Σε κάθε γραμμή οι χαρακτήρες να μην ξεπερνούν κατά πολύ τους 80, ώστε να είναι πάντα ορατή ολόκληρη η γραμμή χωρίς οριζόντια ολίσθηση.
4. Χρήση κενών χαρακτήρων: Πριν και μετά από τους τελεστές να μπαίνουν κενά, ώστε να είναι ευανάγνωστες οι παραστάσεις. Καλύτερα να χρειαστεί δεύτερη και τρίτη γραμμή κώδικα. Οι γραμμές κώδικα μπορούν να χωρίζονται λογικά (πχ ένας όρος αθροίσματος σε κάθε γραμμή).
5. Χρήση κενών γραμμών: Πριν από κάθε «λογική ενότητα» εντολών να μπαίνει μία κενή γραμμή. Κάτω από την κενή γραμμή μπορεί να μπαίνει και ένα σχόλιο που περιγράφει τον σκοπό της ενότητας.
6. Αποφυγή περίπλοκων συναρτήσεων: Κάθε συνάρτηση, να είναι κατά το δυνατόν απλή, ιδανικά να χωράει σε μία «οθόνη» ώστε να φαίνεται ολόκληρος ο κώδικάς της χωρίς κατακόρυφη ολίσθηση. Πριν από κάθε συνάρτηση να εισάγεται και σύντομο σχόλιο.
7. Εύστοχα ονόματα μεταβλητών και συναρτήσεων: Έτσι κάθε όνομα να είναι αυτοεξηγούμενο κατά το δυνατόν. Επίσης να χρησιμοποιείται ένας – πάντα ο ίδιος – τρόπος διαχωρισμού των λέξεων στα περιγραφικά ονόματα. Πχ camelCase ή kebab_case, κλπ
8. Χρήση σταθερών: Αντί για την εμφάνιση «μαγικών» αριθμών. Πχ χρήση του N για το μέγεθος ενός συνόλου, αντί για πχ 18 που είναι, ώστε να είναι εμφανής ο λόγος που πραγματοποιείται αυτό.

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

  intro-hy-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 μεταβλητές

Ο μεταβλητές και οι μηχανισμοί 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;
}
```

Global εμβέλεια μεταβλητών 1/3

Ο πρώτος μηχανισμός είναι η χρήση μεταβλητών έξω από όλες τις συναρτήσεις και – κατά συνέπεια – έξω από όλα τα block εντολών. Αυτές οι μεταβλητές – σε αντίθεση με τις τοπικές (local) – ονομάζονται global (καθολικές) και μπορούν να χρησιμοποιηθούν από οποιοδήποτε σημείο του κώδικα.

Κάποιες εφαρμογές τέτοιων μεταβλητών είναι η αποθήκευση κοινόχρηστης πληροφορίας όπως είναι ρυθμίσεις του προγράμματος, γενικοί μετρητές ή σε άλλη περίπτωση κάποια πληροφορία που θα εξαρτάται από το εκάστοτε πρόβλημα που επιλύεται.

```
let someGlobalCounter = 0;
function func1() {
    someGlobalCounter = ...;
    ...
}
function func2() {
    someGlobalCounter = ...;
    ...
}
```

Global εμβέλεια μεταβλητών 2/3

Η χρήση των μεταβλητών επιτρέπεται από το σημείο της δήλωσής τους και κάτω. Αυτό παίζει ρόλο όταν δηλώσετε μια global μεταβλητή ανάμεσα στις συναρτήσεις. (Σημειώστε ότι στην απλοποιημένη C δεν υπάρχει αυτός ο περιορισμός).

Η αρχική τιμή (όταν αναγράφεται) δίνεται πριν την εκτέλεση της πρώτης εντολής της main και πρέπει να είναι τιμή γνωστή κατά την έναρξη της εκτέλεσης του κώδικά σας. Δεν μπορεί να είναι το αποτέλεσμα της κλήσης μιας συνάρτησης. (Και πάλι σημειώστε ότι στην απλοποιημένη C δεν υπάρχει αυτός ο περιορισμός).

Ενναλακτικά η αρχικοποίησή μιας global μεταβλητής μπορεί να γίνεται σε όποιο σημείο του κώδικα κρίνει κατάλληλο ο προγραμματιστής.

Global εμβέλεια μεταβλητών 3/3

Προσοχή!

Η χρήση Global μεταβλητών δεν αποτελεί καλή πρακτική και πρέπει να γίνεται με μέτρο *εκεί που πραγματικά χρειάζεται*!

Αλλιώς σύντομα θα βρεθείτε να ξοδεύετε τον χρόνο σας στο debugging περίπλοκων σφαλμάτων.

Η χρήση της ελάχιστης δυνατής εμβέλειας των μεταβλητών είναι φίλος του κάθε προγραμματιστή!

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

  intro-hy-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!

Σταθερές τιμές

Κάποιες φορές στον κώδικα πρέπει να χρησιμοποιήσουμε σταθερές τιμές π.χ. το $\boldsymbol{\pi}$ ή το $\boldsymbol{e}$ ή και άλλες – μη μαθηματικές – που εξαρτώνται από το εκάστοτε πρόβλημα που αντιμετωπίζουμε.

Αν και μπορούμε να επαναλαμβάνουμε σε κάθε σημείο του κώδικα τις τιμές αυτές, θα είναι πολύ καλύτερο να έχουμε ονοματισμένη την τιμή ώστε να φαίνεται και τι σημαίνει ο αριθμός αυτός για το πρόβλημά μας.

Η C έχει δύο τρόπους να δημιουργήσει σταθερές, ενώ η C* μόνο τον έναν από αυτούς.

Ο τρόπος αυτός είναι με τη χρήση της λέξης κλειδί **`const`** αντί της **`let`** κατά τη δήλωση μιας μεταβλητής, η οποία ουσιαστικά δεν επιτρέπει άλλη εκχώρηση τιμής στην μεταβλητή αυτή. Π.χ.

```
const N = 10;
```

Δηλαδή δεν έχουμε απλά ονοματίσει τη σταθερή ποσότητα, αλλά έχουμε μία μεταβλητή που την περιέχει και η «μεταβλητή» αυτή είναι «προστατευμένη» από αλλαγές.

Σχεδόν πάντα μια τέτοια σταθερά δηλώνεται έξω από τις συναρτήσεις ως global ώστε όλος ο κώδικάς μας να έχει πρόσβαση σε αυτή.

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

  intro-hy-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!

Διάλειμμα

μικρή ανάσα

Ανέκδοτο της ημέρας

```
// When I wrote this function, only God and I understood
// what it was doing... Now, God only knows
```

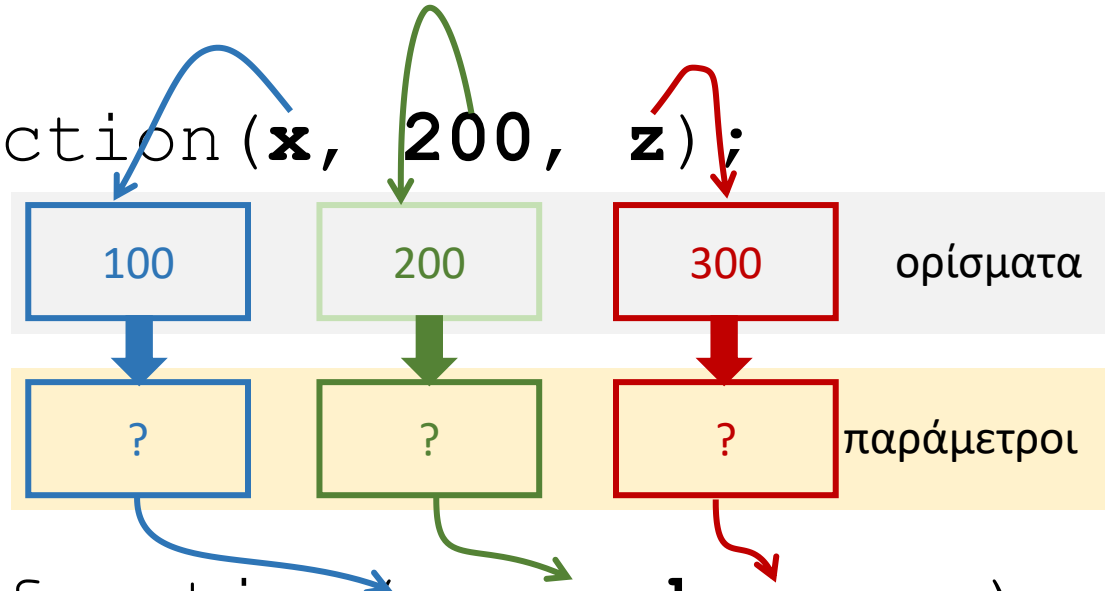
Εμβάθυνση στις συναρτήσεις

Ο μηχανισμός της κλήσης και οι μεταβλητές τους

Call by value – Κλήση με τιμή

Μία κλήση συνάρτησης:

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



Και ο ορισμός της:

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

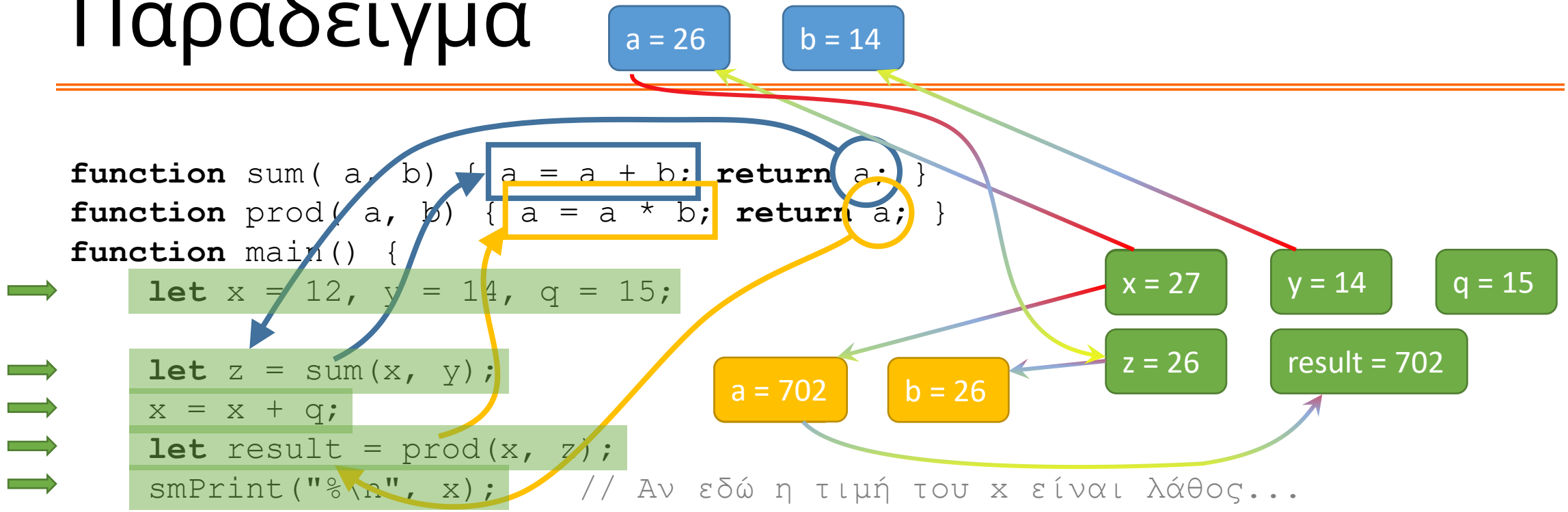
Για να γίνει η κλήση οι τιμές των **ορισμάτων** x,y,z **αντιγράφονται** στις **παραμέτρους** a,b,z και προσέξτε ότι οι a,b,z είναι άλλες μεταβλητές από τις x,y,z !

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

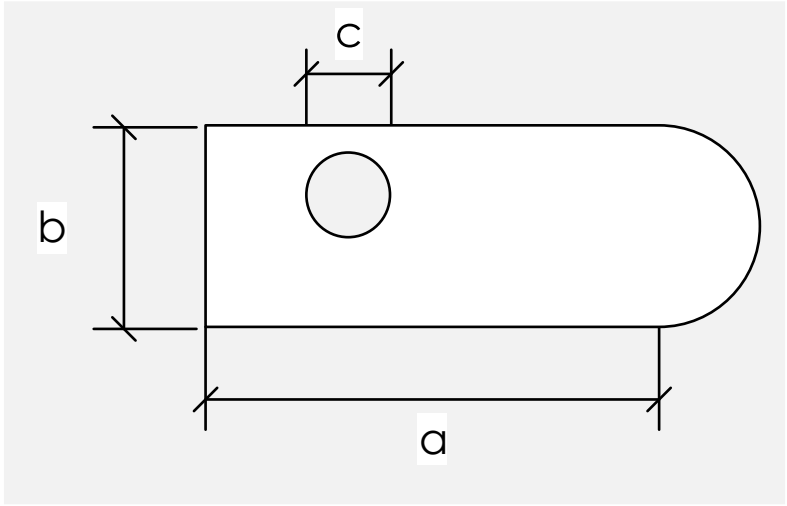
  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  intro-hy-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

  intro-hy-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

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 #2 (σκεπτικό)

1 0

=

D → 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

d → 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

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

  intro-hy-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!