ΜΑΘΗΜΑ 12

ΣΥΝΑΡΤΗΣΕΙΣ ΚΑΙ ΑΝΑΔΡΟΜΗ 1/2

Γιώργος Διάκος - Full Stack Developer

1. Πότε Γράφουμε Συναρτήσεις

- Η γλώσσα C είναι πιο γνωστή και ευρέως χρησιμοποιούμενη **«διαδικαστική»** γλωσσα.
  - Διαδικαστική σημαίνει ότι λειτουργεί με συναρτήσεις (**διαδικασίες**)!
  - Άρα το κύριο χαρακτηριστικό της γλώσσας είναι οι **συναρτήσεις**!

1. Πότε Γράφουμε Συναρτήσεις

- Μία συνάρτηση της C είναι το αντίστοιχο της μαθηματικής συνάρτησης.

  - Θεωρήστε για παράδειγμα την μαθηματική συνάρτηση $f(x) = 5x + 1$
    - Το f είναι το **όνομα** της συνάρτησης
    - Το x ειναι το **όρισμα** της συνάρτησης
    - Το $5x + 1$ είναι το **σώμα** της συνάρτησης

  - Τώρα πως χρησιμοποιούμε μια συνάρτηση.
    - Π.χ. με όρισμα το 2, δηλαδή $f(2)$ (Στην C θα λέμε «**καλώντας την f με όρισμα 2**»)
      - Η συνάρτηση υπολογίζεται: $5 \cdot 2 + 1$
      - $5 \cdot 2 + 1 = 11$ (Στην C θα λέμε «**επιστρέφει 11**»)

    - Π.χ. με όρισμα το 15, δηλαδή το $f(15)$ (Στην C λέμε «καλώντας την f με όρισμα 15»)
      - Η συνάρτηση υπολογίζεται στο $5 \cdot 15 + 1$
      - $5 \cdot 15 + 1 = 76$ (Στην C θα λέμε «επιστρέφει 76»)

1. Πότε Γράφουμε Συναρτήσεις

- Πότε γράφουμε συναρτήσεις;
  - Συχνά όταν γράφουμε ένα μεγάλο πρόγραμμα, υπάρχουν κάποιες ενέργειες που επαναλαμβάνονται.
    - Π.χ. Σε προγράμματα της μετεωρολογίας, απαιτείται συχνά να υπολογιστούν οι λύσεις διαφορικών εξισώσεων
      - Άρα στα προγράμματα τους, έχουν κατασκευάσει γενικές συναρτήσεις που λύνουν διαφορικές εξισώσεις!
    - Σε προγράμματα που χρησιμοποιούνται στις υπηρεσίες π.χ. γραμματειών, γίνονται πολλές φορές οι ίδιες ενέργειες.
      - Έτσι χρησιμοποιούνται συναρτήσεις, για τις επαναλαμβανόμενες ενέργειες εισαγωγής – π.χ. διαγραφής εγγραφών στα δεδομένα που διατηρεί η υπηρεσία.

1. Πότε Γράφουμε Συναρτήσεις

- Οι γενικοί κανόνες που μας καθοδηγούν στο να δημιουργήσουμε μια συνάρτηση στο πρόγραμμά μας είναι:
  - **Γράφουμε συναρτήσεις όταν πολλές φορές στο πρόγραμμα μας κάνουμε τις ίδιες ενέργειες με τον ίδιο κώδικα.**
    - Π.χ. Αν το πρόγραμμα μας κάνει μία εκτύπωση πολλές φορές, τότε θα ορίσουμε μια συνάρτηση με όνομα π.χ. `print()` και καλούμε την συνάρτηση αυτή κάθε φορά που θέλουμε να εκτυπώσουμε τον πίνακα.
  - Και όταν θέλουμε να απλοποιήσουμε την μορφή του προγράμματος μας. **Είναι κακό να έχουμε έναν κώδικα-«μακαρόνι»**, δηλαδή μια τεράστια main που να κάνει πάρα πολλά πράγματα! Προτιμούμε να διασπάμε τον κώδικα σε μέρη και να καλούμε τις αντίστοιχες συναρτήσεις που θα υλοποιούν κάθε αυτόνομη ενέργεια.

- Είδαμε ότι η συνάρτηση θα οριστεί σε 2 σημεία:
  - Στην αρχή του προγράμματος (πριν την main) θα γράψουμε το **πρωτότυπο** της συνάρτησης, που αποτελεί μία απλή περιγραφή των τύπων των δεδομένων των ορισμάτων της και του τύπου του δεδομένου της επιστρεφόμενης τιμής.

  - Αμέσως μετά τη main, ορίζουμε το **σώμα** της συνάρτησης, όπου περιγράφονται οι ενέργειες που εκτελεί η συνάρτηση

- Από την στιγμή που έχουμε ορίσει την συνάρτηση έχουμε το δικαίωμα να την υπολογίσουμε, με συγκεκριμένα ορίσματα, οπουδήποτε μέσα στον κώδικά μας.
  - Η **κλήση** της συνάρτησης είναι να βάλουμε συγκεκριμένα ορίσματα στην συνάρτηση και να την καλέσουμε.

  - Η συνάρτηση θα κάνει τον υπολογισμό της, και θα επιστρέψει το αποτέλεσμά της!

2. Πως Γράφουμε Συναρτήσεις

1. Γενικό Σχήμα

- Ας επαναλάβουμε την γενική εικόνα που θα πρέπει να έχει το πρόγραμμα μας
- Χρησιμοποιούμε στο παράδειγμα αυτό μια πολύ απλή συνάρτηση που δέχεται ως ορίσματα δύο ακεραίους και επιστρέφει το γινόμενο τους.

```
.....

int ginomeno(int x, int y); <- Αυτό είναι το πρωτότυπο της συνάρτησης

main()
{
   ....
   c=ginomeno(a,b); <- Εδώ καλούμε την συνάρτηση στην main,
   ....                σαν μία ακόμη εντολή του προγράμματος
}

int ginomeno(int x, int y) <-Αυτό είναι το σώμα της συνάρτησης
{
   return (x*y);
}
```

2. Πως Γράφουμε Συναρτήσεις

2. Το Πρωτότυπο Συνάρτησης

- ΠΑΝΤΑ πριν από την main καταγράφουμε τα πρωτότυπα των συναρτήσεων που θα ορίσουμε. Το πρωτότυπο είναι μια περιγραφή μόνο των ορισμάτων της συνάρτησης και της επιστρεφόμενης τιμής (και όχι του υπολογισμού). Το συντακτικό είναι:

```
Τύπος_Επιστρεφόμενης_Τιμής ΟΝΟΜΑ_ΣΥΝΑΡΤΗΣΗΣ(Ορισμα1,Ορισμα2,...);
```

- Όπως στην συνάρτηση μάς:

```
int                    ginomeno        (int x,   int y);
```

  - όπου περιγράφουμε ότι πρόκειται να ορίσουμε μια συνάρτηση με όνομα ginomeno: που παίρνει δύο ακέραιες μεταβλητές ως ορίσματα και επιστρέφει μια ακέραια μεταβλητή.

2. Πως Γράφουμε Συναρτήσεις

2. Το Πρωτότυπο Συνάρτησης

- Λίγο πιο αναλυτικά για το πρωτότυπο της συνάρτησης

```
Τύπος_Επιστρεφόμενης_Τιμής ΟΝΟΜΑ_ΣΥΝΑΡΤΗΣΗΣ(Ορισμα1,Ορισμα2,...);
```

- έχουμε:
  - `Τύπος_Επιστρεφόμενης_Τιμής`
    - Μπορεί να είναι οποιοσδήποτε τύπος δεδομένων από όσους μάθαμε στο Μάθημα 3 (π.χ. int, float, double κ.λπ.)
  - `ΟΝΟΜΑ_ΣΥΝΑΡΤΗΣΗΣ`
    - Μπορούμε να δώσουμε οποιοδήποτε όνομα στην συνάρτηση μας, που σέβεται την ονοματολογία που ορίσαμε στο Μάθημα 3 (π.χ. το όνομα δεν μπορεί να ξεκινά με αριθμό)
  - `(Ορισμα1,Ορισμα2,...)`
    - Τα ορίσματα της συνάρτησης χωρίζονται με κόμματα. Κάθε όρισμα έχει την μορφή

      `ΤΔ όνομα_μεταβλητής`
    - Όπου ΤΔ είναι ο τύπος δεδομένων του ορίσματος.

2. Πως Γράφουμε Συναρτήσεις

2. Το Πρωτότυπο Συνάρτησης

- Μερικά Παραδείγματα:
  - `int square(int x);`
  - Είναι μια συνάρτηση με όνομα square που παίρνει σαν όρισμα έναν ακέραιο και θα επιστρέφει έναν ακέραιο αριθμό
  - `double mesos_oros(double x,double y);`
  - Είναι μια συνάρτηση με όνομα mesos_oros που παίρνει ως ορίσματα δύο δεκαδικούς διπλής ακρίβειας και θα επιστρέφει έναν δεκαδικό διπλής ακρίβειας.
  - `void typose_minima(int elegxos);`
  - Είναι μια συνάρτηση που παίρνει σαν όρισμα έναν ακέραιο και **δεν επιστρέφει τίποτα**. Η λέξη `void` στην επιστροφή δηλώνει ότι η συνάρτηση δεν επιστρέφει τίποτα
  - `void ektyposi_plaisiou();`
  - Είναι μια συνάρτηση που δεν παίρνει ορίσματα και δεν επιστρέφει τίποτα! Θα εκτελεί ότι ενέργειες οριστούν στο σώμα της!

2. Πως Γράφουμε Συναρτήσεις

2. Το Πρωτότυπο Συνάρτησης

Συμβουλές:

- Χρησιμοποιούμε **κατατοπιστικά ονόματα** στις συναρτήσεις, ώστε να θυμόμαστε τι ενέργειες εκτελεί!
- Επίσης χρησιμοποιούμε όσο το δυνατόν πιο κατατοπιστικά ονόματα και στα ονόματα των ορισμάτων που δέχεται η συνάρτηση!

- Όταν δηλώνουμε το πρωτότυπο της συνάρτησης δεν ξεχνάμε να βάλουμε ερωτηματικό στο τέλος της δήλωσης!

2. Πως Γράφουμε Συναρτήσεις

3. Το Σώμα Συνάρτησης (Ορισμός)

- Το σώμα της συνάρτησης αποτελεί την περιγραφή των εντολών που εκτελεί η συνάρτηση. Πάντα θα είναι ΜΕΤΑ την main και οι εντολές της θα βρίσκονται ανάμεσα σε άγκιστρα

```
Τύπος_Επιστρεφόμενης_Τιμής ΟΝΟΜΑ_ΣΥΝΑΡΤΗΣΗΣ(Ορισμα1,Ορισμα2,...)
{
  //Εδώ θα δηλώσουμε τις τοπικές μεταβλητές της συνάρτησης

  //Εδώ θα γράψουμε τις εντολές της συνάρτησης
}
```

- Η 1η γραμμή είναι ακριβώς ίδια με το πρωτότυπο (αλλά δεν έχει ερωτηματικό)
- Έπειτα μέσα στα υποχρεωτικά άγκιστρα:
  - Γράφουμε τις τοπικές μεταβλητές που θα χρησιμοποιήσει η συνάρτηση.
  - Και ακολουθούν οι εντολές που θα εκτελέσει η συνάρτηση
- Οι εντολές θα τρέξουν σειριακά (όπως στην main) εωσότου:
  - Είτε φτάσουμε στο τελευταίο άγκιστρο,
  - Είτε φτάσουμε σε μια εντολή return!

Σημείωση:

- Εναλλακτικά η C δίνει τη δυνατότητα να γράψουμε απευθείας το σώμα της συνάρτησης πριν την main και μόνον εκεί. Δεν θα ακολουθήσουμε αυτήν τη προσέγγιση σε αυτές τις σημειώσεις.

2. Πως Γράφουμε Συναρτήσεις

3. Το Σώμα Συνάρτησης (Τοπικές και Καθολικές Μεταβλητές)

- **Τοπικές Μεταβλητές**: Είναι μεταβλητές που δηλώνονται στην αρχή μιας συνάρτησης και τις οποίες τις «βλέπει» (έχει πρόσβαση) η συνάρτηση και ΜΟΝΟΝ αυτή (όχι δηλαδή οι άλλες συναρτήσεις ή η main()
  - Προσοχή! Κάθε συνάρτηση έχει τις δικές της μεταβλητές, έτσι π.χ. μπορούν δύο συναρτήσεις να έχουν μεταβλητές με το ίδιο όνομα. Κάθε συνάρτηση θα «βλέπει» μόνο τις δικές της μεταβλητές.
- **Καθολικές Μεταβλητές:** Είναι μεταβλητές που δηλώνονται πριν από την main και τις οποίες βλέπουν ΟΛΕΣ οι συναρτήσεις (και η main).

- Δείτε το γενικό σχήμα ενός προγράμματος που χρησιμοποιεί τέτοιες μεταβλητές και έπειτα μεταγλωττίστε και εκτελέστε το πρόγραμμα της επόμενης διαφάνειας.

Συμβουλή:

- Θεωρείται **κακή προγραμματιστική τακτική να χρησιμοποιούμε καθολικές μεταβλητές**. Θα πρέπει να γνωρίζουμε πως δουλεύουν, αλλά να μην τις χρησιμοποιούμε στα προγράμματά μας!

2. Πως Γράφουμε Συναρτήσεις

3. Το Σώμα Συνάρτησης (Τοπικές και Καθολικές Μεταβλητές)

```
/* variables.c: Deixnei ton diaxorismo
   katholikwn-topikwn metablitwn*/

#include <stdio.h>

void f1();
void f2();

int x; /* Katholiki metavliti: Tin vlepoun
          oloi */

main()
{
   int a=0; /*Topiki metabliti stin main */

   x=5;
   printf("\nmain: a=%d,x=%d",a,x);
   f1();
   printf("\nmain: a=%d,x=%d",a,x);
   f2();
   printf("\nmain: a=%d,x=%d",a,x);
}
```

```
void f1()
{
   int a=2, x=0; /*Topikes metavlites
                   tis f1 */
   /* Exoyme diplo onoma stin x.
      Epikratei to topiko onoma */

   printf("\nf1: a=%d,x=%d",a,x);
}

void f2()
{
   int a=8; /*Topikes metavlites tis
              f2 */
   x=7; /* Anaferetai stin katholiki
           x */

   printf("\nf2: a=%d,x=%d",a,x);
}
```

ΕΦΑΡΜΟΓΗ 4.

- (α) Χωρίς να εκτελέσεις το πρόγραμμα efarmogi4.c υπολόγισε πόσα Χ θα εκτυπώσει το πρόγραμμα και με ποια μορφή.
- (β) Εκτέλεσε το πρόγραμμα και τροποποίησε το κατά βούληση το "Τρίγωνο " που εκτυπώνεται.
- (γ) Μετάτρεψε το πρόγραμμα σε ενα ισοδύναμο που χρησιμοποιεί την εντολή do...while , αντι για την εντολή for.

ΕΦΑΡΜΟΓΗ 7.

Κατασκεύασε πρόγραμμα που :

- Προτρέπει τον χρήστη να εισάγει εναν αριθμό Ν. Ο αριθμός Ν να είναι μεταξύ 1 και 20 με εφαρμογή αμυντικού προγραμματισμού.
- Έπειτα να διαβάζει από την είσοδο και να εισάγει Ν αριθμούς σε εναν μονοδιάστατο πίνακα.
- Τέλος , να βρίσκει και να τυπώνει το ελάχιστο από τους Ν αριθμούς.

ΥΠΟΔΕΙΞΗ : Όταν δεν ξέρουμε εκ των προτέρων το μέγεθος του πίνακα που θα χρησιμοποιήσουμε , δεσμεύουμε προκαταβολικά τον μέγιστο χώρο που μπορεί το πρόγραμμα να χρησιμοποιήσει.

- **Για παράδειγμα σε αυτή την άσκηση , θα πρέπει να κατασκευάσεις εναν πίνακα 20 θέσεων , παρόλο που το πρόγραμμα θα χρησιμοποιήσει τόσες θέσεις , όσο το Ν που θα εισάγει ο χρήστης.**
- **Σε επόμενο μάθημα θα μάθουμε πως δεσμεύεται ο χώρος στην μνήμη δυναμικά. Δηλαδή κατά τον χρόνο εκτέλεσης να δεσμεύεται ο χώρος μνήμης που απαιτείται.**

ΕΦΑΡΜΟΓΗ 8.

Κατασκεύασε πρόγραμμα που :

- Προτρέπει τον χρήστη να εισάγει εναν αριθμό Ν. Ο αριθμός Ν να είναι μεταξύ 1 και 20 με εφαρμογή αμυντικού προγραμματισμού.
- Έπειτα να διαβάζει από την είσοδο και να εισάγει Ν αριθμούς σε εναν μονοδιάστατο πίνακα.
- Τέλος , να βρίσκει και να τυπώνει το Μ.Ο των Ν αριθμών.

ΤΕΛΟΣ ΜΑΘΗΜΑΤΟΣ

ΣΥΝΑΡΤΗΣΕΙΣ ΚΑΙ ΑΝΑΔΡΟΜΗ 1/2

Γιώργος Διάκος - Full Stack Developer