Η ΓΛΩΣΣΑ C

Μάθημα 6:

Συναρτήσεις και Αναδρομή

Δημήτρης Ψούνης

Περιεχόμενα Μαθήματος

Α. Συναρτήσεις
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**!

Α. 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»)

Α. Συναρτήσεις
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.

Α. Συναρτήσεις
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.

Α. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις

- Το γενικό σχήμα για την σύνταξη μιας συνάρτησης, το είδαμε στο μάθημα 2 «Βασικά Στοιχεία ενός προγράμματος C»
- 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!

Α. Συναρτήσεις
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);
}
```

Α. Συναρτήσεις
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.

Α. Συναρτήσεις
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.

Α. Συναρτήσεις
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!

Α. Συναρτήσεις
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!

Α. Συναρτήσεις
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.

Α. Συναρτήσεις
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!

Α. Συναρτήσεις
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);
}
```


Α. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις
4. Κλήση Συνάρτησης

- Αφού γράψουμε την συνάρτησή μας, έχουμε δικαίωμα να την καλέσουμε οπουδήποτε μέσα στο πρόγραμμα μας. Για να την καλέσουμε:
  - Γράφουμε το όνομα της και διοχετεύουμε κατάλληλα ορίσματα που θα είναι:
    - Είτε απευθείας συγκεκριμένες αριθμητικές τιμές.
    - Είτε ονόματα μεταβλητών που χρησιμοποιούμε ήδη στο πρόγραμμα μας. Προσοχή! Απλά γράφουμε τα ονόματα των μεταβλητών ως ορίσματα και όχι τον τύπο δεδομένων
    - Είτε γενικότερα υπολογιζόμενες παραστάσεις (όπως τις ορίσαμε στο μάθημα 2 που μελετήσαμε τον τελεστή εκχώρησης)

- Δείτε το παράδειγμα της επόμενης διαφάνειας και εντοπίστε στις κλήσεις των συναρτήσεων, να «διοχετεύονται» ορίσματα είτε αριθμητικά, είτε με μεταβλητές που χρησιμοποιεί η «καλούσα συνάρτηση»

Α. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις
4. Κλήση Συνάρτησης

```
/* orismata.c: Anadeikniei pos pairname orismata se sinartiseis */

#include <stdio.h>

int square(int x);

main()
{
   int a=5;
   int b=10;
   int teta,tetb,sum;

   teta=square(a);
   tetb=square(b);
   sum=teta+tetb;
   printf("%d^2 + %d^2 = %d",a,b,sum);
}

int square(int x)
{
  int y;
  y=x*x;
  return y;
}
```


Α. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις
4. Κλήση Συνάρτησης

- Όπως φαίνεται και από το παράδειγμα, η επιστρεφόμενη τιμή της συνάρτησης γίνεται με την εντολή return:

```
return τιμή;
```

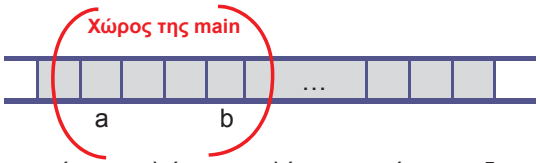
  - Η δεσμευμένη λέξη return ακολουθείται από την τιμή, η οποία μπορεί να είναι οποιαδήποτε υπολογιζόμενη παράσταση (σταθερά, μεταβλητή, υπολογισμός, ή ακόμη και συνάρτηση)
  - Σημαντικό: Η εκτέλεση της εντολής return σταματά επιτόπου την εκτέλεση της συνάρτησης, υπολογίζει την τιμή και επιστρέφει στην καλούσα συνάρτηση.

- Ισχύει επίσης ότι σε μια συνάρτηση που επιστρέφει void, μπορούμε να σταματήσουμε την εκτέλεση της με την εντολή:

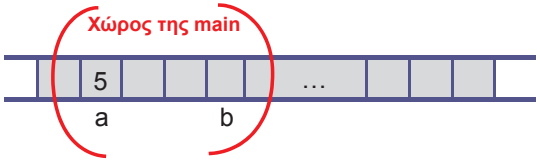
```
return;
```


Α. Συναρτήσεις
3. Πως Λειτουργούν οι Συναρτήσεις
1. Συναρτήσεις και Χώρος στη Μνήμη

- Είναι σημαντικό να καταλάβουμε ότι κάθε συνάρτηση έχει το δικό της «χώρο» στη μνήμη, στον οποίο αποθηκεύει τις μεταβλητές της.
- Για παράδειγμα έστω το τμήμα κώδικα που φαίνεται στα δεξιά
- Όταν ξεκινάει να εκτελείται ο κώδικας υπάρχει ο χώρος αποθήκευσης μόνο για την main!



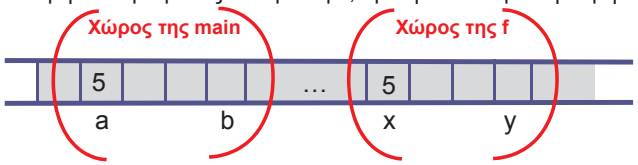
- Έπειτα όταν εκτελείται η εντολή αρχικοποίησης a=5, η κατάσταση της μνήμης είναι:



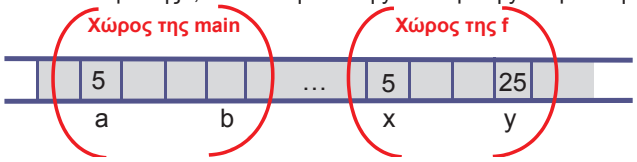

```
int f(int x);

main()
{
  int a=5,b;
  b=f(a);
}

int f(int x)
{
  int y;
  y=x*x;
  return y;
}
```


Α. Συναρτήσεις
3. Πως Λειτουργούν οι Συναρτήσεις
1. Συναρτήσεις και Χώρος στη Μνήμη

- Έπειτα καλείται η f με όρισμα a.
  - Προσοχή! Αυτό σημαίνει, ότι δημιουργείται χώρος αποθήκευσης για την f.
  - Και στον χώρο αποθήκευσης της f, η μεταβλητή x θα πάρει την τιμή του ορίσματος που βάλαμε, άρα η x θα πάρει την τιμή 5.



  - Είναι σημαντικό να καταλάβουμε ότι από εδώ και πέρα η x δεν έχει καμία σχέση με την a. Έχει τον δικό της χώρο μνήμης και την διαχειρίζεται η f.
- Πλέον στο σώμα της f, καλείται η εντολή y=x*x άρα η y παίρνει την 25



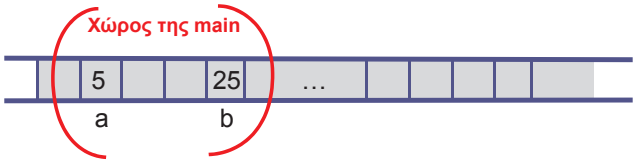

```
int f(int x);

main()
{
  int a=5,b;
  b=f(a);
}

int f(int x)
{
  int y;
  y=x*x;
  return y;
}
```

A. Συναρτήσεις

3. Πως Λειτουργούν οι Συναρτήσεις

1. Συναρτήσεις και Χώρος στη Μνήμη

- Καλείται η εντολή return y. Αυτό σημαίνει ότι επιστρέφουμε στην main!
  - Πλέον η επιστρεφόμενη τιμή (25) αποθηκεύεται στην μεταβλητή b.
  - Είναι σημαντικό ότι μετά την επιστροφή τιμής ο χώρος της f, απελευθερώνεται για να μπορεί να χρησιμοποιηθεί από άλλες συναρτήσεις:




```
int f(int x);

main()
{
  int a=5,b;
  b=f(a);
}

int f(int x)
{
  int y;
  y=x*x;
  return y;
}
```

- Το παράδειγμα αυτό αναδεικνύει δύο σημαντικά θέματα!
  - Κάθε κλήση συνάρτησης δημιουργεί τον δικό της χώρο στην μνήμη!
  - Ο μόνος δίαυλος επικοινωνίας με την καλούσα συνάρτηση είναι τα ορίσματα (στην αρχή) και η επιστρεφόμενη τιμή (στο τέλος)

A. Συναρτήσεις

3. Πως Λειτουργούν οι Συναρτήσεις

2. Περίπλοκα Ορίσματα

- Στην πραγματικότητα η διοχέτευση ορίσματος:
  - είναι απλά, να αρχικοποιηθεί η τιμή του ορίσματος με την τιμή που δίνουμε από την καλούσα συνάρτηση!
- Έτσι έχουμε το δικαίωμα να θέσουμε ως ορίσματα οποιαδήποτε παράσταση της C
  - Τότε απλά θα έχουμε ότι θα υπολογιστεί η τιμή του ορίσματος, και έπειτα η μεταβλητή του ορίσματος θα πάρει την κατάλληλη τιμή.
- Για παράδειγμα στο τμήμα κώδικα:

```
int a=5;
int b;
b=f(2*a);
```

  - Υπολογίζεται πρώτα το όρισμα (είναι η τιμή 10)
  - Έπειτα καλείται η f με όρισμα την τιμή 10

- Ενώ αν η f είναι η συνάρτηση που είδαμε προηγουμένως τότε έχουμε δικαίωμα να γράψουμε ακόμη και το εξής:

```
int a=5;
int b;
b=f(f(a));
```

  - Όπου υπολογίζεται πρώτα το εσωτερικό f(a) (υπολογίζεται σε 25) και μετά υπολογίζεται το f(25) άρα αποθηκεύεται στο b η τιμή 625.

A. Συναρτήσεις

3. Πως Λειτουργούν οι Συναρτήσεις

3. Παραπάνω του ενός ορίσματα

- Αντίστοιχοι είναι οι κανόνες όταν έχουμε μια συνάρτηση που δέχεται πολλά ορίσματα. Πχ. Αν έχουμε ορίσει την συνάρτηση:

```
void func(int x, int y, int z)
```

- Και την καλέσουμε ως εξής:

```
func(1,5,10)
```

  - Τότε η τιμή που θέσαμε πρώτη αποθηκεύεται στην μεταβλητή του 1ου ορίσματος (δηλαδή το x παίρνει την τιμή 1)
  - Η τιμή που θέσαμε δεύτερη αποθηκεύεται στην μεταβλητή του 2ου ορίσματος (δηλαδή το y παίρνει την τιμή 5)
  - Η τιμή που θέσαμε τρίτη αποθηκεύεται στην μεταβλητή του 3ου ορίσματος (δηλαδή το z παίρνει την τιμή 10)

B. Αναδρομή

1. Κλήση Συνάρτησης μέσα σε Συνάρτηση

- Όπως είδαμε αφού ορίσουμε μια συνάρτηση μπορούμε να την καλέσουμε οπουδήποτε στον κώδικα.
- Άρα μπορούμε να καλέσουμε μια συνάρτηση που έχουμε ορίσει μέσα σε μια άλλη συνάρτηση!
- Π.χ. ας ορίσουμε μια συνάρτηση που υπολογίζει την $f(x)=2x^2$

```
int f(int x)
{
  int y;
  y=2*x*x;
  return y;
}
```

- Μπορούμε να ορίσουμε την συνάρτηση $g(x)=2x^2+x+1$ ως εξής:

```
int g(int x)
{
  int y;
  y=f(x)+x+1;
  return y;
}
```

Β. Αναδρομή
2. Αναδρομικές Συναρτήσεις

- Ο όρος **αναδρομή αναφέρεται σε μια συνάρτηση που καλεί τον εαυτό της**!
  - Αυτό είναι απόλυτα νόμιμο και στην C, αφού απλά καλούμε μια συνάρτηση μέσα σε μια συνάρτηση!
- Άρα μια συνάρτηση που στο σώμα της καλεί τον εαυτό της, θα ονομάζεται **αναδρομική συνάρτηση.**
- Η δημιουργία μιας αναδρομικής συνάρτησης είναι πολύ χρήσιμη, ιδίως όταν κατασκευάζουμε πράγματα που ορίζονται αναδρομικά!

- Ας δούμε ένα παράδειγμα:
  - Το παραγοντικό του φυσικού αριθμού n ορίζεται ως:
    - $n! = n \cdot (n-1) \cdot (n-2) \cdot \cdots 2 \cdot 1$
    - π.χ. έχουμε $1! = 1,\ \ 2! = 2 \cdot 1,\ \ 3! = 3 \cdot 2 \cdot 1 = 6,\ \ 4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$ κ.ο.κ.
  - Το παραγοντικό ορίζεται ωστόσο και αναδρομικά ως εξής:
    - $n! = n \cdot (n-1)!$ αν $n > 1$
    - $n! = 1,$ αν $n = 1$

Β. Αναδρομή
2. Αναδρομικές Συναρτήσεις
1. Υπολογισμός Παραγοντικού

```
/* factorial.c: Ypologizei to paragontiko
   enos fysikoy */

#include <stdio.h>

int factorial(int n);

main()
{
   int x;
   int res;

   printf("Dwste ton fysiko: ");
   scanf("%d",&x);

   res=factorial(x);

   printf("%d!=%d",x,res);
}
```

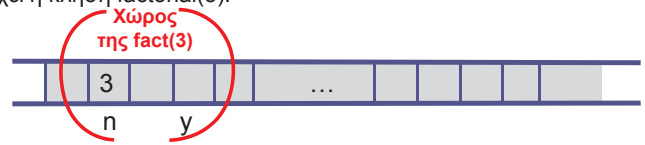
```
int factorial(int n)
{
  int y;

  if (n==1)
    return 1;
  else
  {
    y=factorial(n-1);
    return n*y;
  }
}
```



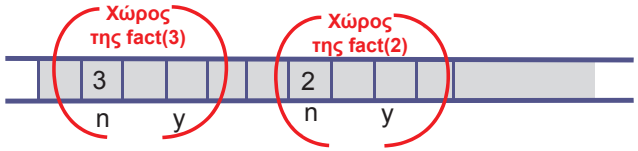
```
int fact (int n)
{
  int y;
  if (n==1) return 1;
  else{
    y=factorial(n-1);
    return n*y;
  }
}
```

Β. Αναδρομή
2. Αναδρομικές Συναρτήσεις
1. Υπολογισμός Παραγοντικού

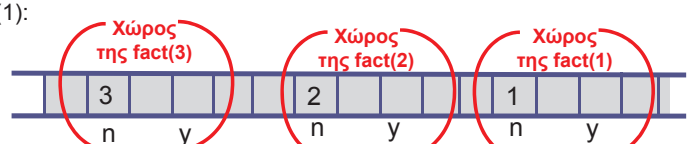
- Στον υπολογισμό μίας αναδρομικής συνάρτησης, κάθε αναδρομική κλήση έχει και το δικό της χώρο στη μνήμη.
- Ας δούμε πως τρέχει η κλήση factorial(3):



- Καλεί την factorial(2):



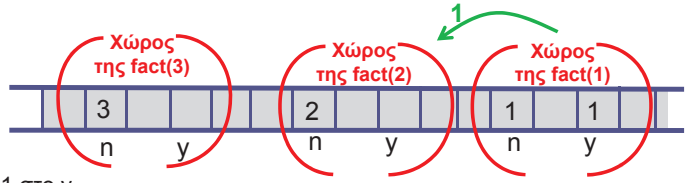
- Καλεί την factorial(1):



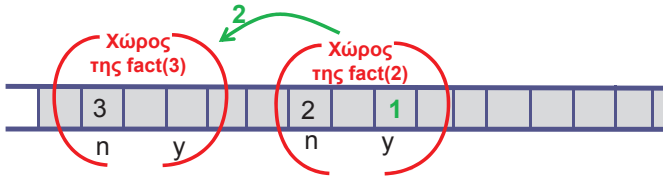

```
int fact (int n)
{
  int y;
  if (n==1) return 1;
  else{
    y=factorial(n-1);
    return n*y;
  }
}
```

Β. Αναδρομή
2. Αναδρομικές Συναρτήσεις
1. Υπολογισμός Παραγοντικού

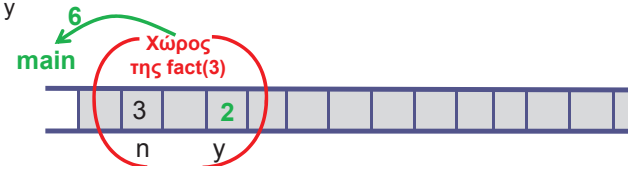
- Η factorial(1) επιστρέφει 1:



- Η factorial(2) αποθηκεύει το 1 στο y
  - έπειτα επιστρέφει 2*1.



- Η factorial(2) αποθηκεύει το 1 στο y
  - έπειτα επιστρέφει 3*2



Β. Αναδρομή
2. Αναδρομικές Συναρτήσεις
1. Υπολογισμός Παραγοντικού

```
int fact (int n)
{
  int y;
  if (n==1) return 1;
  else{
    y=factorial(n-1);
    return n*y;
  }
}
```

- Για να απεικονίσουμε τις αναδρομικές κλήσεις που γίνονται προτιμάται μία παράσταση όπου κάθε αναδρομική κλήση στοιχίζεται λίγο δεξιότερα.
- Με τον τρόπο αυτό μπορούμε να παρακολουθήσουμε αρκετά ικανοποιητικά την εκτέλεση ενός αναδρομικού κώδικα:

```
ΚΛΗΣΗ fact(3)
(3==1) ΟΧΙ
y=fact(2)
    ΚΛΗΣΗ fact(2)
    (2==1) ΟΧΙ
    y=fact(1)
        ΚΛΗΣΗ fact(1)
        (1==1) ΝΑΙ
        return 1
    y=1
    return 2*1=2
y=2
return 3*2=6
```


Γ. Ασκήσεις
Εφαρμογή 1 (Συναρτήσεις Ελέγχου Εισόδου)

- Ορίστε την συνάρτηση:
  - `int get_integer(int start, int finish)`: Θα λαμβάνει ως είσοδο ένα εύρος τιμών ακεραίων `[start…finish]` και θα διαβάζει έναν ακέραιο σε αυτό το εύρος. Θα επιστρέφει τον αριθμό που διαβάστηκε.
- Ορίστε τη συνάρτηση `main` να διαβάζει δύο ακέραιους `a,b` στο διάστημα 1..10 και έναν ακέραιο `n` στο διάστημα 2..5 και θα υπολογίζει την ποσότητα n*(a-b) και θα χρησιμοποιεί τη συνάρτηση που ορίσαμε.

Γ. Ασκήσεις
Εφαρμογή 2 (Μια Βιβλιοθήκη Μελέτης Αριθμών)

- Ορίζουμε τις συναρτήσεις:
  - `int is_even(int n)`: Θα επιστρέφει 0 ή 1 ανάλογα με το αν ο αριθμός είναι άρτιος
  - `int is_odd(int n)`: Θα επιστρέφει 0 ή 1 ανάλογα με το αν ο αριθμός είναι περιττός
  - `int is_square(int n)`: Θα επιστρέφει 0 ή 1 ανάλογα με το αν ο αριθμός είναι τετράγωνο ενός φυσικού
  - `int is_cube(int n)`: Θα επιστρέφει 0 ή 1 ανάλογα με το αν ο αριθμός είναι κύβος ενός φυσικού
- Ορίζουμε την main που θα ζητάει από το χρήστη είτε να εισάγει έναν αριθμό και θα εξετάζει αν ο αριθμός έχει κάποιες από αυτές τις ιδιότητες.
- Παράδειγμα Εκτέλεσης:

```
Eisagete ton arithmo: 8
Einai Artios
Einai Kivos Arithmou
```

```
Eisagete ton arithmo: 9
Einai Perittos
Einai Tetragono Arithmou
```


Γ. Ασκήσεις
Εφαρμογή 3 (Πρώτοι Αριθμοί)

- Ένας φυσικός αριθμός λέμε ότι είναι πρώτος αν διαιρείται (ακριβώς) μόνο με τον εαυτό του και τη μονάδα. Το 1 θεωρείται ότι δεν είναι πρώτος.
- Κατασκευάστε ένα πρόγραμμα το οποίο:
  - Θα ορίζει μία συνάρτηση με όνομα isprime() η οποία θα δέχεται ως όρισμα έναν ακέραιο αριθμό, θα εξετάζει αν είναι πρώτος και θα επιστρέφει 1 αν είναι πρώτος και 0 αν δεν είναι.
  - Η main θα διαβάζει δύο φυσικούς (ελέγχοντας στην είσοδο να είναι >0) που θα ορίζουν την αρχή και το τέλος ενός κλειστού διαστήματος (π.χ. a=5, b=8) και θα τυπώνει τους φυσικούς σε αυτό το διάστημα που είναι πρώτοι.
- Παράδειγμα εκτέλεσης του ζητούμενου προγράμματος:

```
Eisagete tin arxi tou diastimatos: 5
Eisagete to peras tou diastimatos: 15

To 5 einai prwtos
To 7 einai prwtos
To 11 einai prwtos
To 13 einai prwtos
```

Γ. Ασκήσεις

Εφαρμογή 4 (Αναδρομή: Η ακολουθία FIbonacci)

- Η ακολουθία fibonacci ορίζεται ως:
  - $F_n=F_{n-1}+F_{n-2}$, για n>2
  - $F_2=1$
  - $F_1=1$
- Για παράδειγμα έχουμε $F_1=1,F_2=1,F_3=2,F_4=3,F_5=5,F_6=8$ κ.ο.κ.
- Ορίστε την συνάρτηση int fibonacci(int n) που δέχεται ως όρισμα έναν φυσικό και επιστρέφει το n-οστό fibonacci.
- Έπειτα κατασκευάστε μία main που διαβάζει από τον χρήστη έναν ακέραιο και υπολογίζει και επιστρέφει τον αριθμό fibonacci του αριθμού που εισήγαγε ο χρήστης.

Γ. Ασκήσεις

Εφαρμογή 5 (Αναδρομή: ΜΚΔ με τον αλγόριθμο του Ευκλείδη)

- Ο αλγόριθμος του Ευκλείδη για την εύρεση του Μέγιστου Κοινού Διαιρέτη δύο (φυσικών) αριθμών:
  - Ξεκινά με ένα ζεύγος φυσικών και σχηματίζει ένα νέο ζευγάρι με τον μικρότερο αριθμό και την διαφορά του μικρότερου από τον μεγαλύτερο αριθμό.
  - Η διαδικασία επαναλαμβάνεται εωσότου οι αριθμοί γίνουν ίσοι. Ο αριθμός αυτός είναι ο ΜΚΔ των αρχικών αριθμών.
- Μαθηματικά ο ΜΚΔ(a,b) όπου a,b είναι φυσικοί:
  - Είναι ίσο με a, αν a=b
  - Είναι ίσο με ΜΚΔ(a,b-a), αν a<b
  - Είναι ίσο με ΜΚΔ(a-b,b), αλλιώς
- Κατασκευάστε ένα πρόγραμμα σε γλώσσα C που θα υλοποιεί με μία αναδρομική συνάρτηση τον υπολογισμό του ΜΚΔ και μία συνάρτηση main που θα ζητάει από το χρήστη να εισάγει τους δύο φυσικούς, θα κάνει κατάλληλη κλήση της συνάρτησης και θα τυπώνει τον ΜΚΔ των αριθμών.