ΥΠΟΛΟΓΙΣΤΕΣ ΙΙ

Συναρτήσεις

Τι είναι ;

- Αυτόνομα τμήματα κώδικα (υποπρογράμματα) που πραγματοποιούν μια καθορισμένη εργασία.
- Χρήσιμες για περιπτώσεις που ο ίδιος υπολογισμός επαναλαμβάνεται πολλές φορές μέσα στο πρόγραμμα.
- Βοηθούν στην τμηματική ανάπτυξη και έλεγχο μεγάλων προγραμμάτων.
- Παράδειγμα: οι ενσωματωμένες συναρτήσεις sin, cos, sqrt, abs, κλπ
- Κάθε συνάρτηση μπορεί να δέχεται ως είσοδο μία ή περισσότερες μεταβλητές και μπορεί να επιστρέφει ένα αποτέλεσμα (ή παραπάνω).
- Κάθε πρόγραμμα C++ αποτελείται από μία η περισσότερες συναρτήσεις.

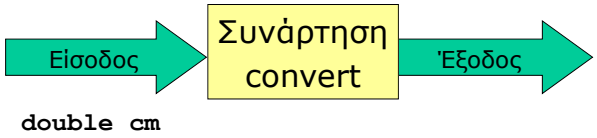
Παράδειγμα #1

Κατασκευάστε συνάρτηση που θα δέχεται εκατοστά και θα τα μετατρέπει σε ίντσες.

Υπενθύμιση:
1 in = 2.54 cm

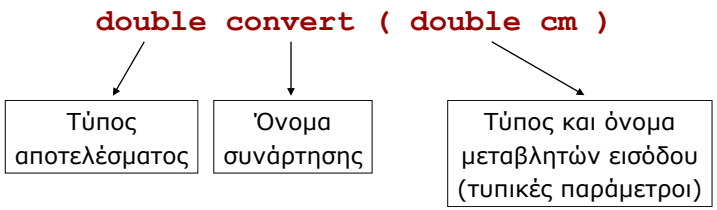
Παράδειγμα #1

```
double convert ( double cm )
{
    double inch;

    inch = cm/2.54;
    return inch;
}
```



Παράδειγμα #1

Η επικεφαλίδα της συνάρτησης:



Παράδειγμα #1 (πλήρες)

```
#include <iostream>
using namespace std;

double convert ( double );              // Πρωτότυπο συνάρτησης

int main ( )
{
    double a, b;

    cout << "Πόσα εκατοστά ? ";
    cin >> a;

    b = convert(a);                     // Κλήση συνάρτησης
    cout << a << " εκατοστά είναι " << b << " ίντσες \n";
}

double convert ( double cm )
{
    double inch;

    inch = cm/2.54;                     // Κώδικας συνάρτησης
    return inch;
}
```


Η μορφή μιας συνάρτησης

τύπος αποτελέσματος *όνομα συνάρτησης* (*παράμετροι*)
{
    δηλώσεις τοπικών μεταβλητών
    ... εντολές ...
    return *τιμή αποτελέσματος*
}

Η δομή ενός προγράμματος με συναρτήσεις

```
δηλώσεις #include
πρωτότυπα συναρτήσεων
int main ( )
{
    … εντολές …
}
```

συνάρτηση1

συνάρτηση2

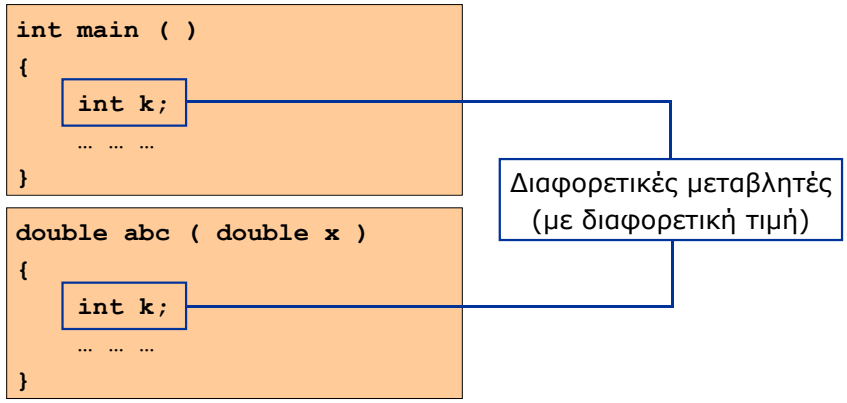
Το κυρίως πρόγραμμα main είναι μια συνάρτηση που καλείται από το λειτουργικό σύστημα.

Το πρωτότυπο της συνάρτησης

- Χρειάζεται έτσι ώστε ο μεταφραστής να γνωρίζει τι τύπου ορίσματα δέχεται η συνάρτηση και τι τύπου αποτέλεσμα επιστρέφει.
- Είναι ίδιο με την επικεφαλίδα της συνάρτησης χωρίς τα ονόματα των μεταβλητών.
- Τα πρωτότυπα γράφονται πριν από το main αλλά γενικά πριν χρησιμοποιήσουμε τις συναρτήσεις.
- Στο τέλος του πρωτότυπου μπαίνει ερωτηματικό.

Μεταβλητές της συνάρτησης

Οι μεταβλητές που ορίζονται εσωτερικά σε μια συνάρτηση αποτελούν τοπικές μεταβλητές της συνάρτησης και δεν έχουν σχέση με μεταβλητές που έχουν το ίδιο όνομα σε άλλες συναρτήσεις.



Τι τύπου αποτέλεσμα επιστρέφει μια συνάρτηση ;

- Μια συνάρτηση μπορεί να επιστρέψει αποτέλεσμα οποιουδήποτε από τους τύπους δεδομένων που χειρίζεται η γλώσσα (double, int, κλπ).
- Ο τερματισμός λειτουργίας της συνάρτησης και η επιστροφή του αποτελέσματος γίνεται με την εντολή return.

Κλήση και επιστροφή της συνάρτησης

- Για να χρησιμοποιήσουμε μια συνάρτηση πρέπει να την *καλέσουμε*. Πχ. η συνάρτηση
  ```
  double convert ( double cm )
  ```
  καλείται ως:
  ```
  b = convert(a)
  ```
- Οι μεταβλητές στην επικεφαλίδα της συνάρτησης (cm) ονομάζονται *τυπικές παράμετροι*.
- Οι μεταβλητές που δίνουμε κατά την κλήση (a) ονομάζονται *πραγματικές παράμετροι* ή *ορίσματα* της κλήσης. Το πλήθος και οι τύποι των ορισμάτων κατά την κλήση πρέπει να συμβαδίζουν με αυτά που δηλώθηκαν στο πρωτότυπο και την επικεφαλίδα της συνάρτησης.
- Κάθε συνάρτηση μπορεί να κληθεί πολλές φορές με διαφορετικά ορίσματα. Πχ:
  ```
  x = convert(y)
  q = convert(3*w-t)
  ```

Κλήση και επιστροφή της συνάρτησης

Η κλήση και η επιστροφή της συνάρτησης γίνεται ως εξής:

- Οι πραγματικές παράμετροι αντιγράφονται στις τυπικές παραμέτρους. Πχ αν η συνάρτηση

```
double convert ( double cm )
```

κληθεί ως:

```
b = convert(a)
```

τότε η τιμή της a αντιγράφεται στην cm. Ο τρόπος αυτός μεταβίβασης τιμών ονομάζεται μεταβίβαση κατ' αξία.

- Ο υπολογιστής εγκαταλείπει το σημείο που βρισκόταν και πηγαίνει στον κώδικα της συνάρτησης.
- Εκτελούνται οι εντολές της συνάρτησης και μόλις βρεθεί η εντολή return, ο υπολογιστής επιστρέφει στο σημείο από όπου κλήθηκε η συνάρτηση.

Παράδειγμα #2

Κατασκευάστε συνάρτηση που θα δέχεται ως είσοδο μια γωνία σε ακτίνια και θα την μετατρέπει σε μοίρες.

Υπενθύμιση:

Μοίρες = Ακτίνια*180/π

Παράδειγμα #2

```
#include <iostream>
#include <cmath>
using namespace std;

double moires ( double );                    // Πρωτότυπο

int main ( )
{
    double a;

    cout << "Πόσα ακτίνια ? ";
    cin >> a;

    cout << a << " ακτίνια είναι " << moires(a)
         << " μοίρες \n";
}

double moires ( double rad )
{
    return rad*180/acos(-1.0);
}
```


Παράδειγμα #3

Υλοποιήστε σε C++ τη συνάρτηση:

$$f(x) = \frac{7x^2 - 3x + 6}{\sqrt{1 + x^2}}$$

Παράδειγμα #3

```
double f ( double x )
{
    return (7*x*x-3*x+6)/sqrt(1+x*x);
}
```


Παράδειγμα #4

Εμφανίστε στην οθόνη πίνακα τιμών της προηγούμενης συνάρτησης, από x=0 έως 10 ανά 0.1

Παράδειγμα #4

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

double f ( double );

int main ( )
{
    double x;

    cout << fixed;
    for (x=0; x<=10; x+=0.1)
        cout << setw(5)  << setprecision(1) << x << "   "
             << setw(10) << setprecision(7) << f(x)
             << endl;
}
```


Παράδειγμα #4

Αποτέλεσμα στην οθόνη:

```
  0.0    6.0000000
  0.1    5.7413646
  0.2    5.5696982
  0.3    5.4883446
  0.4    5.4965820
  0.5    5.5901699
  0.6    5.7623525
  0.7    6.0049700
  0.8    6.3094200
  0.9    6.6673485
  1.0    7.0710678
  1.1    7.5137551
  1.2    7.9895013
  1.3    8.4932709
  1.4    9.0208168
  1.5    9.5685784
   …        …
```

Συναρτήσεις που δεν επιστρέφουν αποτέλεσμα

Μια συνάρτηση μπορεί να μην επιστρέφει αποτέλεσμα. Λέμε ότι είναι συνάρτηση τύπου void.

Μια συνάρτηση που δεν επιστρέφει αποτέλεσμα δεν καλείται με τον ίδιο τρόπο όπως οι άλλες συναρτήσεις, αλλά η κλήση της μοιάζει με εντολή.

Παράδειγμα #5

Κατασκευάστε συνάρτηση που θα δέχεται ως είσοδο έναν ακέραιο n και θα εμφανίζει στην οθόνη n φορές τη φράση:
Κατανοώ τη γλώσσα C++

Παράδειγμα #5

```
#include <iostream>
using namespace std;

void katanoo ( int );                          // Πρωτότυπο συνάρτησης

int main ( )
{
    int k;

    cout << "Πόσες φορές ? ";
    cin >> k;

    katanoo(k);                                  // Κλήση συνάρτησης
}

void katanoo ( int n )
{
    int i;

    for (i=1; i<=n; ++i)
       cout << "Κατανοώ τη γλώσσα C++ \n"
}
```

// Κώδικας συνάρτησης

Συναρτήσεις χωρίς παραμέτρους

Υπάρχουν συναρτήσεις που εκτελούν μια συγκεκριμένη εργασία χωρίς να δέχονται παραμέτρους.

Παράδειγμα #6

Κατασκευάστε συνάρτηση που θα υπολογίζει και θα επιστρέφει το λόγο της χρυσής τομής

$$\frac{\sqrt{5}-1}{2}$$

Παράδειγμα #6

```
double golden ( )
{
    return (sqrt(5.0)-1)/2;
}
```

Το αντίστοιχο πρωτότυπο θα δηλωθεί ως:

```
double golden ( void );
```

Η κλήση της συνάρτησης:

```
a = golden();
```


Παράδειγμα #7

Κατασκευάστε συνάρτηση που θα δέχεται ως είσοδο τις τρεις καρτεσιανές συντεταγμένες ενός διανύσματος και θα υπολογίζει το μέτρο του.

Παράδειγμα #7

```
#include <iostream>
#include <cmath>
using namespace std;

double metro ( double, double, double );

int main ( )
{
    double x, y, z;

    cout << "Εισάγετε συντεταγμένες ";
    cin >> x >> y >> z;

    cout << "Το μέτρο είναι " << metro(x,y,z) << endl;
}

double metro ( double x, double y, double z)
{
    return sqrt(x*x+y*y+z*z);
}
```

Ποιος μπορεί να καλέσει μια συνάρτηση ;

Μια συνάρτηση μπορεί να κληθεί είτε από το κυρίως πρόγραμμα main ή από οποιαδήποτε άλλη συνάρτηση.

Παράδειγμα #8

Κατασκευάστε τη συνάρτηση

```
double gonia ( double x1, double y1, double z1,
               double x2, double y2, double z2 )
```

που θα υπολογίζει τη γωνία (σε μοίρες) μεταξύ δύο διανυσμάτων r_1, r_2 με συντεταγμένες (x_1,y_1,z_1) και (x_2,y_2,z_2).

Υπενθύμιση:

$$\cos\varphi = \frac{\vec{r}_1\vec{r}_2}{|\vec{r}_1||\vec{r}_2|} = \frac{x_1x_2 + y_1y_2 + z_1z_2}{\sqrt{x_1^2 + y_1^2 + z_1^2}\sqrt{x_2^2 + y_2^2 + z_2^2}}$$

Κατασκευάστε πρώτα μια συνάρτηση για το εσωτερικό γινόμενο και χρησιμοποιείστε τις συναρτήσεις metro και moires που φτιάξαμε πριν.

Παράδειγμα #8

```
double ginomeno ( double x1, double y1, double z1,
                  double x2, double y2, double z2 )
{
    return x1*x2 + y1*y2 + z1*z2;
}
```


Παράδειγμα #8

```
double gonia ( double x1, double y1, double z1,
               double x2, double y2, double z2 )
{
    double m1, m2;
    double g, t, phi, mphi;

    m1 = metro(x1,y1,z1);
    m2 = metro(x2,y2,z2);
    if ( m1*m2 == 0 )
        mphi = 0;
    else {
        g = ginomeno(x1,y1,z1,x2,y2,z2);
        t = g/(m1*m2);
        if ( t > 1 )  t = 1;
        if ( t < -1 ) t = -1;
        phi = acos(t);
        mphi = moires(phi);
    }
    return mphi;
}
```

Παράδειγμα #8

Τι χρειάζονται οι εντολές:

```
if ( t > 1 )  t = 1;
if ( t < -1 ) t = -1;
```

Κάποιες φορές λόγω σφαλμάτων στρογγύλευσης μπορεί το t (δηλαδή το cosφ) να υπολογιστεί ως:
1.0000000000000001

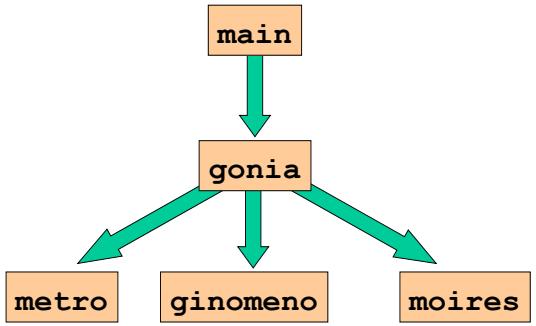
Σε αυτή την περίπτωση η κλήση της συνάρτησης acos που ακολουθεί θα δώσει λάθος.

Παράδειγμα #8

```
#include <iostream>
#include <cmath>
using namespace std;

double metro ( double,double,double );
double ginomeno ( double,double,double,double,double,double );
double gonia ( double,double,double,double,double,double );
double moires ( double );

int main ( )
{
    double x1, y1, z1, x2, y2, z2;

    cout << "Εισάγετε συντεταγμένες ";
    cin >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;

    cout << "Η γωνία είναι "
         << gonia(x1,y1,z1,x2,y2,z2) << endl;
}
```


Παράδειγμα #8

Πως καλούνται οι συναρτήσεις στο παράδειγμα.



Παράδειγμα #9

Δίνεται η συνάρτηση

$$f(x)=\frac{x^2+2x+6}{\sqrt{x^2+1}}$$

Υπολογίστε μια προσέγγιση της πρώτης παραγώγου, από τον ορισμό

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x-h)}{2h}$$

χρησιμοποιώντας ένα μικρό h. Κατόπιν υπολογίστε την παράσταση

$$\Phi=\frac{2f(x)+f'(x)}{3f(x)-f'(x)}$$

για οποιοδήποτε x ζητηθεί.

Παράδειγμα #9

Θα κατασκευάσουμε τρεις συναρτήσεις:

- Για τη συνάρτηση f(x)
  `double f ( double x )`
- Για την παράγωγο f'(x)
  `double deriv ( double x )`
- Για την παράσταση Φ
  `double phi ( double x )`

Παράδειγμα #9

```
double f ( double x )
{
    return (x*x+2*x+6)/sqrt(1+x*x);
}

double deriv ( double x )
{
    double h;

    if ( x == 0 )
        h = 1.0e-5;
    else
        h = 1.0e-5*x;
    return (f(x+h)-f(x-h))/(2*h);
}
```


Παράδειγμα #9

```
double phi ( double x )
{
    double fx, dx;

    fx = f(x);
    dx = deriv(x);
    return (2*fx+dx)/(3*fx-dx);
}
```

Γιατί δεν γράψαμε το παρακάτω ;

```
return (2*f(x)+deriv(x))/(3*f(x)-deriv(x));
```


Παράδειγμα #9

```
#include <iostream>
#include <cmath>
using namespace std;

double f ( double );
double deriv ( double );
double phi ( double );

int main ( )
{
    double x;

    cout << "Ποιό x ? ";
    cin >> x;

    cout << phi(x) << endl;
}
```

Τοπικές και καθολικές μεταβλητές

- Κάθε συνάρτηση έχει τις δικές της μεταβλητές (τοπικές μεταβλητές).
- Οι τοπικές μεταβλητές ισχύουν μόνο μέσα στη συνάρτηση στην οποία δηλώθηκαν και για όσο λειτουργεί η συνάρτηση.
- Όταν η συνάρτηση τερματίσει τη λειτουργία της οι τοπικές μεταβλητές καταστρέφονται και χάνουν τα περιεχόμενά τους.
- Μπορούμε να δηλώσουμε μεταβλητές που δεν ανήκουν σε καμία συνάρτηση (καθολικές μεταβλητές).
- Οι καθολικές μεταβλητές μπορούν να χρησιμοποιηθούν από όλες τις συναρτήσεις και τα περιεχόμενά τους παραμένουν καθ' όλη τη διάρκεια του προγράμματος.

Δήλωση καθολικών μεταβλητών

Οι καθολικές μεταβλητές δηλώνονται εκτός οποιασδήποτε συνάρτησης.

```
#include <iostream>
using namespace std;

double x, y, z;            // Καθολικές μεταβλητές

int main ( )
{
    … εντολές …
}

double add ( double a, double b )
{
    … εντολές …
}
```


Καθολικές μεταβλητές

- Εάν υπάρχει σε μια συνάρτηση δηλωμένη τοπική μεταβλητή με το ίδιο όνομα με κάποια καθολική μεταβλητή, τότε στη συνάρτηση αυτή χρησιμοποιείται η τοπική και όχι η καθολική μεταβλητή.
- Κατά συνέπεια η συνάρτηση αυτή δεν μπορεί να έχει πρόσβαση στην αντίστοιχη καθολική μεταβλητή.
- Με τη χρήση καθολικών μεταβλητών μπορούμε να δώσουμε μεταβιβάσουμε τιμές στις συναρτήσεις, κάτι όμως που αν γίνεται συστηματικά οδηγεί σε δυσνόητα προγράμματα με δυσκολία στην εκσφαλμάτωση.

Αναδρομικότητα

- Μια συνάρτηση μπορεί να καλέσει το εαυτό της. Όταν συμβαίνει αυτό έχουμε μια αναδρομική κλήση της συνάρτησης.
- Αναδρομική κλήση μπορούμε να έχουμε όταν μια συνάρτηση καλέσει τον εαυτό της μέσω κάποιας άλλης συνάρτησης. Πχ
  Η συνάρτηση a καλεί την b
  Η συνάρτηση b καλεί την c
  Η συνάρτηση c καλεί την a
- Στην πραγματικότητα δεν καλείται πάλι η ίδια συνάρτηση, αλλά ένα δεύτερο (ή τρίτο…) αντίγραφό της, το οποίο δημιουργείται αυτόματα όταν χρειαστεί.

Παράδειγμα #10

Κατασκευάστε τη συνάρτηση

`int parag ( int n )`

που θα υπολογίζει το n παραγοντικό

$$n! = 1 \cdot 2 \cdot 3 \cdots n$$

Παράδειγμα #10

Με τον παραδοσιακό (όχι αναδρομικό τρόπο)

```
int parag ( int n )
{
    int p, k;

    p = 1;
    for (k=2; k<=n; ++k)
        p*=k;
     return p;
}
```


Παράδειγμα #10

Χρησιμοποιώντας αναδρομικότητα:
Παρατηρούμε ότι n! = n (n-1)!

```
int parag ( int n )
{
    if ( n > 1 )
        return n*parag(n-1);
    else
        return 1;
}
```


Υπερφόρτωση συναρτήσεων

- Η C++ επιτρέπει τον ορισμό συναρτήσεων με το ίδιο όνομα, αλλά διαφορετικές παραμέτρους και λειτουργία, στο ίδιο πρόγραμμα. Η τεχνική αυτή ονομάζεται υπερφόρτωση συναρτήσεων.
- Οι γνωστές μαθηματικές συναρτήσεις είναι ήδη υπερφορτωμένες αφού μπορούν να κληθούν με ορίσματα τύπου float ή double. Τα αντίστοιχα πρωτότυπα για την sqrt θα ήταν:

  `float sqrt ( float )`
  `double sqrt ( double )`

- Κατά την κλήση της συνάρτησης, ανάλογα με τον τύπο των ορισμάτων ο μεταφραστής αποφασίζει ποια από τις διαφορετικές εκδοχές της συνάρτησης θα καλέσει στην πραγματικότητα.

Παράδειγμα #11

Κατασκευάστε συνάρτηση η οποία θα υπολογίζει το υπόλοιπο της διαίρεσης του *a* από τον *b*. Η συνάρτηση θα πρέπει να καλείται είτε με ακέραια ορίσματα ή με πραγματικούς διπλής ακρίβειας.

Θα κατασκευάσουμε δύο διαφορετικές συναρτήσεις με το ίδιο όνομα οι οποίες όμως θα δέχονται διαφορετικούς τύπους παραμέτρων.

Παράδειγμα #11

Για την περίπτωση ακέραιων

```
int mod ( int a, int b )
{
        return a % b;
}
```


Παράδειγμα #11

Για την περίπτωση πραγματικών διπλής ακρίβειας

```
double mod ( double a, double b )
{
        int p;
        double y;

        p = (int) (a/b);            // Ακέραιο πηλίκο a/b
        y = a - p*b;                // Το υπόλοιπο
        return y;
}
```


Τελεστής αλλαγής τύπου

- Με τον τελεστή `(τύπος)` μπροστά από μια παράσταση αλλάζουμε την παράσταση στον δεδομένο τύπο. Ο τύπος μπορεί να είναι int, float, double, char κλπ.
- Έτσι με τον τελεστή `(int)` αλλάζουμε μια παράσταση σε ακέραια. Τα δεκαδικά ψηφία αποκόπτονται.
- Η αλλαγή τύπου έχει υψηλότερη προτεραιότητα από τις πράξεις. Πχ.

  `(int)x+0.5`

  είναι διαφορετικό από το

  `(int)(x+0.5)`
- Όταν αλλάζουμε μια παράσταση σε κατώτερο τύπο (πχ. από double σε float) χάνουμε σε ακρίβεια.

Παράδειγμα #11

Το κυρίως πρόγραμμα

```
#include <iostream>
using namespace std;

int mod ( int, int );              // Πρωτότυπα
double mod ( double, double );

int main ( )
{
    int k;
    double d;

    k = mod(25,7);                 // Κλήση με ακέραιους
    cout << k << endl;

    d = mod(7.4,2.2);              // Κλήση με double
    cout << d << endl;
}
```