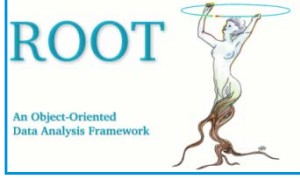
Συναρτήσεις και Πίνακες

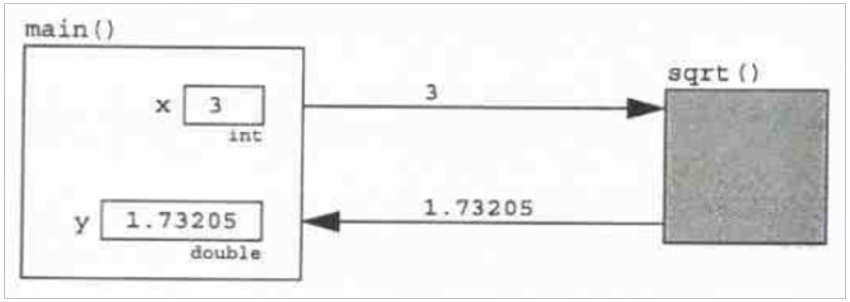
- Συναρτήσεις καθιερωμένης βιβλιοθήκης της C++
- Συναρτήσεις οριζόμενες από τον χρήστη
- Μεταβίβαση κατ' αξία
- Συναρτήσεις void και λογικές συναρτήσεις
- Μεταβίβαση κατ' αναφορά
- Επιστροφή περισσοτέρων από μιας τιμών
- Εμβόλιμες συναρτήσεις
- Υπερφόρτωση συναρτήσεων
- Η συνάρτηση main()
- Μονοδιάστατοι Πίνακες
- Πολυδιάστατοι Πίνακες
- Η συνάρτηση assert() και το typedef

Συναρτήσεις καθιερωμένης βιβλιοθήκης της C++



- Η *καθιερωμένη βιβλιοθήκη της C++* είναι μια συλογή από προκαθορισμένες συναρτήσεις και άλλα προγραμματιστικά στοιχεία τα οποία προσπελάζονται μέσω των αρχείων-κεφαλίδων (headers files).
- Παράδειγμα : Κλίση συνάρτησης sqrt()

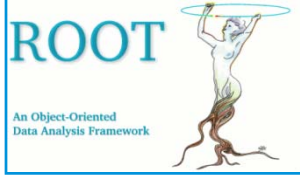
```
#include <cmath>
.....
double x=3., y;
y=sqrt(x);
```



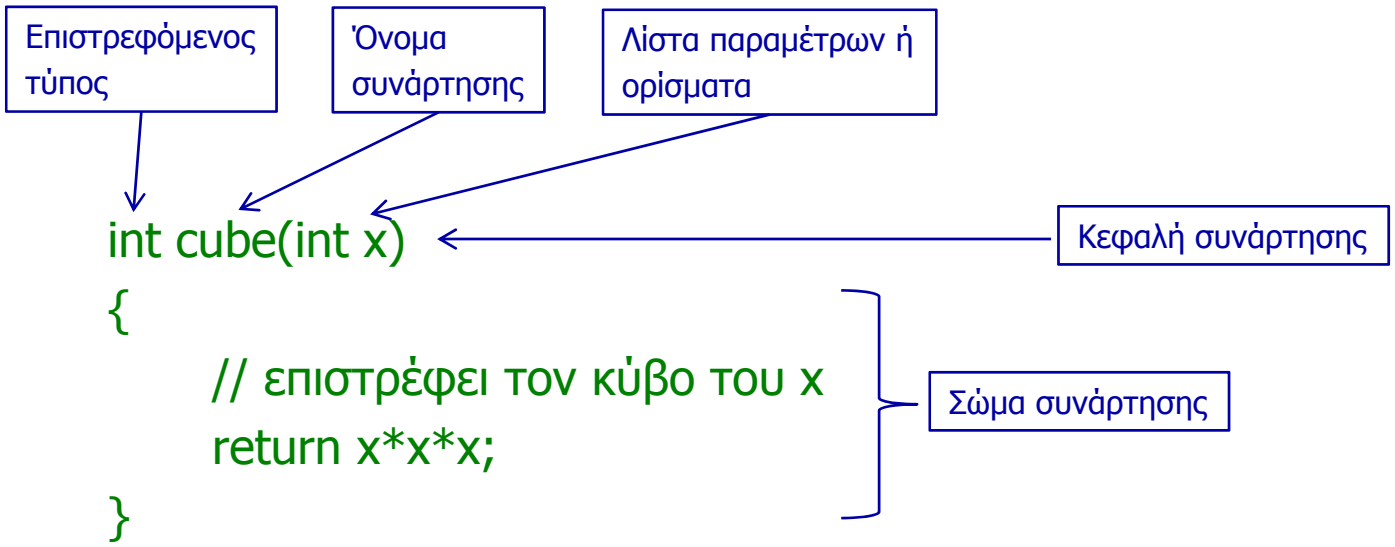
Μερικά αρχεία-κεφαλίδες της Καθιερωμένης βιβλιοθήκης της C++

Αρχείο κεφαλίδας	Περιγραφή
<cassert>	Ορίζει τη συνάρτηση assert()
<ctype>	Ορίζει συναρτήσεις για τη δοκιμή χαρακτήρων
<cfloat>	Ορίζει σταθερές σχετικές με αριθμούς τύπου float
<climits>	Ορίζει τα όρια των ακεραίων στο τοπικό σας σύστημα
<cmath>	Ορίζει μαθηματικές συναρτήσεις
<cstdio>	Ορίζει συναρτήσεις για την καθιερωμένη είσοδο και έξοδο
<cstdlib>	Ορίζει βοηθητικές συναρτήσεις
<cstring>	Ορίζει συναρτήσεις για την επεξεργασία αλφαριθμητικών
<ctime>	Ορίζει συναρτήσεις χρόνου και ημερομηνίας

Συναρτήσεις οριζόμενες από τον χρήστη

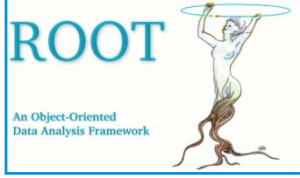


- Μια συνάρτηση οριζόμενη από τον χρήστη έχει δύο μέρη: την *κεφαλή* και το *σώμα*.

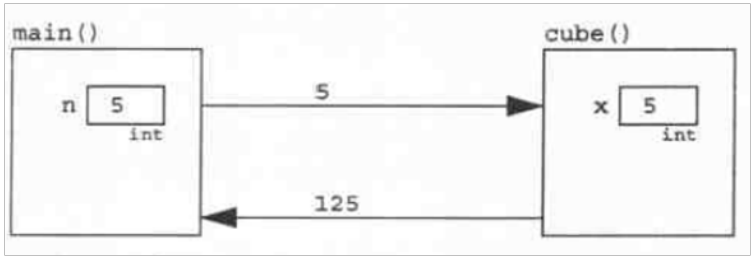


- Η εντολή return μιας συνάρτησης εξυπηρετεί δύο σκοπούς: τερματίζει την εκτέλεση της συνάρτησης και επιστρέφει μια τιμή στο καλούν πρόγραμμα.

Μεταβίβαση κατ' αξία



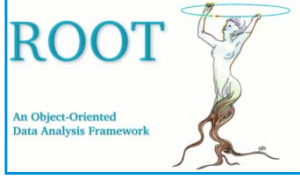

```
//  Δοκιμαστής για τη συνάρτηση cube()
#include <iostream>
using namespace std;

int cube(int x);          // Δήλωση της συνάρτησης

int main()
{// Η συνατηση main() καλεί την συνάρτηση cube()
  int n=1;
  while (n != 0)
  { cin >> n;
    cout << "\tcube(" << n << ") = " << cube(n) << endl;
  }
}

int cube(int x)      // Ο ορισμός της συνάρτησης cube()
{ // returns cube of x:
  return x*x*x;
}
```



Η συνάρτηση main() μεταβιβάζει στην cube() την τιμή 5, και η cube() επιστρέφει στη main() την τιμή 125. Το όρισμα n μεταβιβάζεται κατ' αξία (by value) στην τυπική παράμετρο x.

Αυτό απλά σημαίνει πως, κατά την κλήση της συνάρτησης, εκχωρείται στην x η τιμή του n.

Συναρτήσεις void και λογικές συναρτήσεις

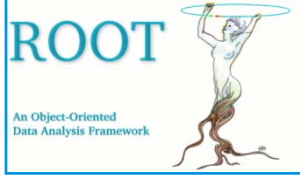


- Όταν μια συνάρτηση δεν επιστρέφει κάποια τιμή έχει τύπο void. Πχ. όταν μια συνάρτηση τυπώνει απλά κάποιες πληροφορίες.

```
//  Κατάταξη χαρακτήρων
#include <cctype>    // Ορίζει τις λογικές συναρτήσεις isdigit() etc.
#include <iostream>
using namespace std;

void printCharCategory(char c);      // Δήλωση της συνάρτησης

int main() {
  for (int c=0; c < 128; c++)
    printCharCategory(c);
}

void printCharCategory(char c)
{ // prints the category to which the given character belongs:
  cout << "The character [" << c << "] is a ";
  if      (isdigit(c)) cout << "digit.\n";
  else if (islower(c)) cout << "lower-case letter.\n";
  else if (isupper(c)) cout << "capital letter.\n";
  else if (isspace(c)) cout << "white space character.\n";
  else if (iscntrl(c)) cout << "control character.\n";
  else if (ispunct(c)) cout << "punctuation mark.\n";
  else                 cout << "Error.\n";
}
```

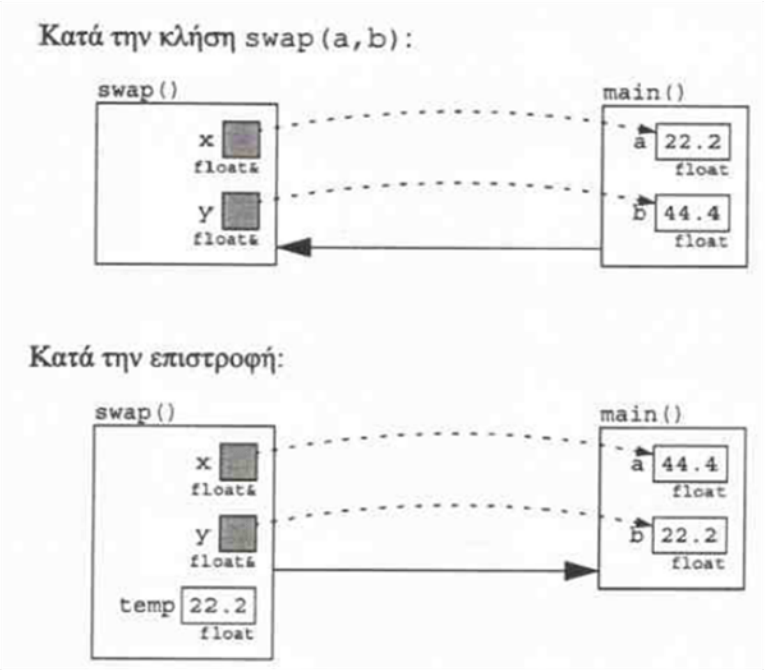
Οι συναρτήσεις isdigit(), islower() κτλ. ονομάζονται και Λογικές Συναρτήσεις (Boolean Functions).

Μεταβίβαση κατ' αναφορά

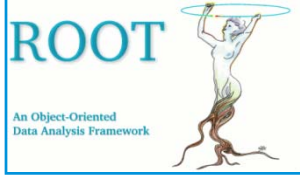


- Υπάρχουν πολλές περιπτώσεις στον προγραμματισμό όπου η συνάρτηση πρέπει να αλλάξει την τιμή της παραμέτρου η οποία μεταβιβάζεται σε αυτή. Αυτό γίνεται με την μεταβίβαση κατ' αναφορά (by reference).

```
//  Η συνάρτηση swap()
#include <iostream>
using namespace std;

void swap(float&, float&); // Δήλωση της συνάρτησης

int main()
{ // tests the swap() function:
  float a = 22.2, b = 44.4;
  cout << "a = " << a << ", b = " << b << endl;
  swap(a,b);
  cout << "a = " << a << ", b = " << b << endl;
}

void swap(float& x, float& y)
{ // Εναλλάσσει τις τιμές των x και y
  float temp = x;
  x = y;
  y = temp;
}
```



Μεταβίβαση κατ' αναφορά

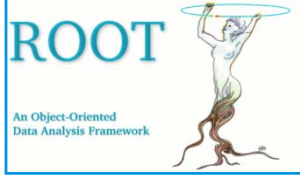


- Για να μεταβιβαστεί μια παράμετρος κατ' αναφορά αντί κατ' αξία, προστίθεται το σύμβολο & στο προσδιοριστικό του τύπου στη λίστα παραμέτρων της συνάρτησης.
- Αυτό κάνει την τοπική μεταβλητή μια αναφορά του ορίσματος που μεταβιβάζεται σε αυτή.
- Το όρισμα γίνεται ανάγνωσης-εγγραφής (read-write) αντί μόνο για ανάγνωση.
- Οποιαδήποτε αλλαγή στην τοπική μεταβλητή στο εσωτερικό της συνάρτησης θα προκαλέσει την ίδια αλλαγή στο όρισμα που μεταβιβάστηκε σε αυτή.

Σύγκριση μεταβίβασης κατ' αξία και μεταβίβασης κατ' αναφορά

Μεταβίβαση κατ' αξία	Μεταβίβαση κατ' αναφορά
`int x;`	`int &x;`
Η παράμετρος x είναι τοπική μεταβλητή.	Η παράμετρος x είναι τοπική αναφορά.
Είναι αντίγραφο του ορίσματος.	Είναι συνώνυμο του ορίσματος.
Δεν μπορεί να αλλάξει το όρισμα.	Μπορεί να αλλάξει το όρισμα.
Το όρισμα που μεταβιβάζεται κατ' αξία μπορεί να είναι σταθερά, μεταβλητή ή μια παράσταση.	Το όρισμα που μεταβιβάζεται κατ' αναφορά πρέπει να είναι μεταβλητή.
Το όρισμα είναι μόνο για ανάγνωση.	Το όρισμα είναι για ανάγνωση-εγγραφή.

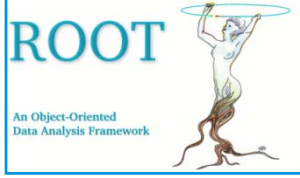
Επιστροφή περισσοτέρων από μιας τιμών



- Η παρακάτω συνάρτηση επιστρέφει δύο τιμές χρησιμοποιώντας δύο παραμέτρους αναφοράς: το εμβαδό και την περιφέρεια ενός κύκλου ακτίνας r.

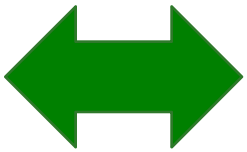
```
//  Επιστροφή περισσότερων από μίας τιμών
#include <iostream>
using namespace std;
void computeCircle(double&, double&, double);     // Δήλωση της συνάρτησης

int main(){
  double r, a, c;
  cout << "Enter radius: ";
  cin >> r;
  computeCircle(a, c, r);
  cout << "area = " << a << ", circumference = " << c << endl;
}

void computeCircle(double& area, double& circumference, double r)
{ // Επιστρέφει το εμβαδό και την περιφέρεια ενός κύκλου ακτίνας r
  const double PI = 3.141592653589793;
  area = PI*r*r;
  circumference = 2*PI*r;
}
```

Εμβόλιμες συναρτήσεις



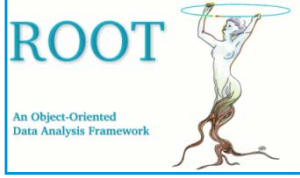
- Όταν μια συνάρτηση χαρακτηρίζεται ως inline (εμβόλιμη) αυτό σημαίνει πως ο μεταγλωττιστής πρέπει να αντικαταστήσει κάθε κλήση της συνάρτησης με ρητό κώδικα.

```
//  Μετατροπή της συνάρτησης Cube σε
      εμβόλιμη
#include <iostream>
using namespace std;
inline int cube(int);   // Δήλωση
 int main()
{ // tests the cube() function:
  cout << cube(4) << endl;
  int x, y;
  cin >> x;
  y = cube(2*x-3);
}
inline int cube(int x)
{ // returns cube of x:
  return x*x*x;
}
```



```
//  Ισοδύναμος κώδικας

#include <iostream>
using namespace std;


int main()
{ // Ισοδύναμος κώδικας
  cout << 4*4*4 << endl;
  int x, y;
  cin >> x;
  y = (2*x-3)* (2*x-3) * (2*x-3);
}
```

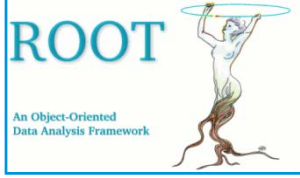
Υπερφόρτωση



- Η C++ επιτρέπει τη χρήση του ιδίου ονόματος για διαφορετικές συναρτήσεις αρκεί να έχουν διαφορετικό αριθμό παραμέτρων είτε να υπάρχει μια θέση στη λίστα παραμέτρων όπου οι τύποι να είναι διαφορετικοί.

```
//  Υπερφόρτωση της συνάρτησης max()
#include <iostream>
using namespace std;
int max(int, int);        // Δήλωση της max() με δύο ορίσματα
int max(int, int, int);  // Δήλωση της max() με τρία ορίσματα

int main(){
    cout << max(99,77) << " " << max(55,66,33);

}

int max(int x, int y)
{ // Επιστρέφει τον μέγιστο δύο ακεραίων
     return (x > y ? x : y);

}

int max(int x, int y, int z)
{ // Επιστρέφει τον μέγιστο τριών ακεραίων
     int m = (x > y ? x : y);
     return (z > m ? z : m);
}
```

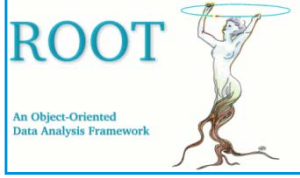
Η συνάρτηση main()



- Η συνάρτηση main() αποτελεί την βασική συνάρτηση όπου ξεκινά και τελειώνει η εκτέλεση κάθε προγράμματος.
- Συνήθως ο τύπος της είναι int και τελειώνει με το return 0;
- Ο τερματισμός ενός προγράμματος μπορεί να γίνει:
  - Με την χρήση της εντολής return στη main()
  - Με τη χρήση των συναρτήσεων exit() ή abort()
  - Μεταβίβαση μιας εξαίρεσης που δεν εντοπίστηκε

```
//  Χρήση της συνάρτησης exit() για τον τερματισμό ενός προγράμματος
#include <cstdlib>   // Ορίζει τη συνάρτηση exit()
#include <iostream>
using namespace std;
double reciprocal(double x);

int main(){
    double x;
    cin >> x;
    cout << reciprocal(x);
}

double reciprocal(double x){
    // returns the reciprocal of x:
    if (x == 0) exit(1);    // Τερματισμός του προγράμματος
    return 1.0/x;
}
```

Μονοδιάστατοι Πίνακες



- Ο *πίνακας* (array) είναι μία ακολουθία αντικειμένων τα οποία έχουν όλα τον ίδιο τύπο.
- Παραδείγματα:

```
int a[10];        // Πίνακας ακεραίων με δέκα στοιχεία
double x[20];     // Πίνακας διπλής ακρίβειας με είκοσι στοιχεία
char c[15];       // Πίνακας χαρακτήρων με δεκαπέντε στοιχεία
```

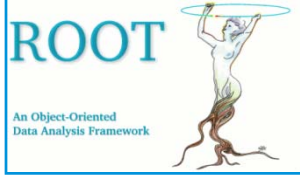
- Η αρίθμηση των στοιχείων ενός πίνακα γίνεται με βάση το μηδέν:

  int a[10] ⟹ a[0], a[1], a[2], a[3] ..... a[8], a[9]

- Η απόδοση αρχικών τιμών σε ένα πίνακα μπορεί να γίνει ως :
  - int a[5] = {15, 17, 6, 89, 20};
  - int a[] = {15, 17, 6, 89, 20};
  - int a[5];

```
a[0]=15;
a[1]=17;
.....
a[4]=20;
```

Μονοδιάστατοι Πίνακες



- Ο αριθμός των στοιχείων ενός πίνακα μπορεί να υπολογιστεί χρησιμοποιώντας την συνάρτηση sizeof() η οποία επιστρέφει το μέγεθος ενός αντικειμένου σε bytes:

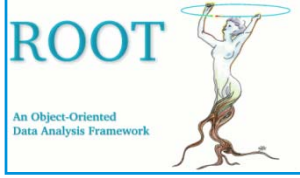
  int a[8]; ⟶ sizeof(a)/sizeof(int)

  double x[12]; ⟶ sizeof(x)/sizeof(double)

- Η μεταβίβαση ενός πίνακα σε συνάρτηση γίνεται όπως στο ακόλουθο παράδειγμα:

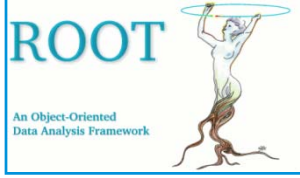
```
//  Μεταβίβαση πίνακα σε συνάρτηση που επιστρέφει το άθροισμα των στοιχείων του
#include <iostream>
using namespace std;
int sum(int a[], int n);   // Δήλωση της sum() που υπολογίζει το άθροισμα των στοιχείων

int main() { // tests using array parameters:
    int a[] = { 11, 33, 55, 77 };
    int size = sizeof(a)/sizeof(int);
    cout << "sum(a,size) = " << sum(a,size) << endl;
}

int sum(int a[], int n) { // Επιστρέφει το άθροισμα των n πρώτων στοιχείων του πίνακα a[]
  int sum=0;
  for (int i=0; i<n; i++)
       sum += a[i];
  return sum;
}
```

Παράδειγμα: Ο Αλγόριθμος Ταξινόμησης Φυσαλίδας




```
//  Ταξινόμηση φυσαλίδας
#include <iostream>
using namespace std;
void print(float[],int);        // Δήλωση της print() εκτύπωση του πίνακα
void swap(float&,float&);   // Δήλωση της swap() εναλλαγή τιμών
void sort(float[],int);         // Δήλωση της sort() ταξινόμηση του πίνακα

int main(){
    float a[] = {55.5, 22.5, 99.9, 66.6, 44.4, 88.8, 33.3, 77.7};
    print(a,8);
    sort(a,8);
    print(a,8);
}

void print(float a[], int n){
    for (int i=0; i<n; i++){ cout << "a[" << i << "]=" << a[i] << endl;}
}

void swap(float& x, float& y) { // Εναλλάσσει τις τιμές των x και y
  float temp = x;
  x = y;
  y = temp;
}

void sort(float a[], int n){    // Ταξινόμηση Φυσαλίδας
    for (int i=1; i<n; i++)
        for (int j=0; j<n-i; j++)
            if (a[j] > a[j+1]) swap(a[j],a[j+1]);
}
```

Πολυδιάστατοι Πίνακες



- Στη C++ μπορούμε να ορίσουμε πίνακες πολλών διαστάσεων. Για παράδειγμα ο πίνακας:

  int a[][]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};

  αντιστοιχεί στον

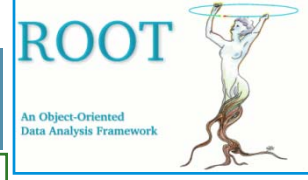
  1 2 3 4
  5 6 7 8
  9 10 11 12

  δηλαδή

  a[0][0] = 1; a[1][2] = 7;
  a[0][1] = 2; a[1][3] = 8;
  a[0][2] = 3; a[2][0] = 9;
  a[0][3] = 4; a[2][1] = 10;
  a[1][0] = 5; a[2][2] = 11;
  a[1][1] = 6; a[2][3] = 12;

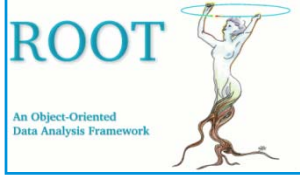
- Ακολουθεί παράδειγμα ανάγνωσης και εκτύπωσης δισδιάστατου πίνακα

Πολυδιάστατοι Πίνακες




```
//  Ανάγνωση και εκτύπωση δισδιάστατου πίνακα
#include <iostream>
using namespace std;
void read(int a[][5]);
void print(const int a[][5]);

int main(){
    int a[3][5];
    read(a);
    print(a);
}

void read(int a[][5]) {
    cout << "Enter 15 integers, 5 per row:\n";
    for (int i=0; i<3; i++) {
        cout << "Row " << i << ": ";
        for (int j=0; j<5; j++)
            cin >> a[i][j];
    }
}

void print(const int a[][5]) {
    for (int i=0; i<3; i++) {
        for (int j=0; j<5; j++)
            cout << " " << a[i][j];
        cout << endl;
   }
}
```

Η χρήση της συνάρτησης assert()

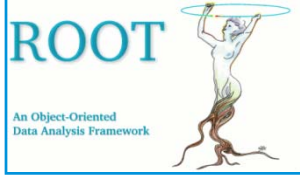


- Η συνάρτηση assert(), η οποία εμπεριέχεται στο αρχείο-κεφαλίδα <cassert>, χρησιμοποιείται όταν θέλουμε να επιβάλουμε μια προαπαιτούμενη συνθήκη.

```
// Test assert() function
#include<cassert>  // Define assert()
#include <iostream>
using namespace std;
float min(float[],int);     // Evaluates min of array

int main(){
  float a[] = {55.5, 22.5, 99.9, 66.6, 44.4, 88.8, 33.3, 77.7};
  cout << "min=" << min(a,8) << endl;
  cout << "min=" << min(a,-8) << endl;
}

float min(float a[], int n){
  assert(n>=0);
  float min=a[0];
  for(int i=0; i<n; ++i)
    if(a[i] < min) min=a[i];
  return min;
}
```

```
[panos@pc-247 Cpp]$ c++ test_assert.cpp
[panos@pc-247 Cpp]$ a.out
min=22.5
a.out: test_assert.cpp:14: float min(float*, int): Assertion `n>=0' failed.
Aborted
[panos@pc-247 Cpp]$
```

Ορισμοί τύπων χρησιμοποιώντας το typedef

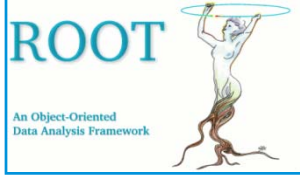


- Η λέξη κλειδί typedef δηλώνει ένα νέο όνομα (δηλαδή ένα συνώνυμο ή ψευδώνυμο) ενός συγκεκριμένου τύπου. Η σύνταξή του είναι:

  typedef *τύπος ψευδώνυμο ;*

```
//  Παραδείγματα χρήσης του typedef
#include <iostream>
using namespace std;

typedef long Integer;          // Integer συνώνυμο του long
typedef double Real; ;         // Real συνώνυμο του double
typedef float Sequence[];      // Sequence συνώνυμο του πίνακα float

int main() {
    Integer n=123;
    Real x=3.14;
    const Real PI=3.141592653589;
    Sequence a[]={5.2, 7.8, 9.1, 1.2, 7.7};
     ..........
     ..........
}
```

Απλό Παράδειγμα με συνάρτηση



- Παράδειγμα συνάρτησης η οποία υπολογίζει και επιστρέφει τον μέγιστο μεταξύ δύο ακεραίων αριθμών.

```
//  Η συνάρτηση max()
#include <iostream>
using namespace std;

int max(int,int);     // Δήλωση της συνάρτησης

int main()
{ // Η συνατηση main() καλεί την συνάρτηση max()
  int m, n;
  do
{ cin >> m >> n;
        cout << "\tmax(" << m << "," << n << ") = " << max(m,n) << endl;
  }
  while (m != 0);
}

int max(int x, int y)     // Ο ορισμός της συνάρτησης max()
{ if (x < y) return y;
  else return x;
}
```