Πανεπιστήμιο Αιγαίου

Μεθοδολογίες και Γλώσσες Προγραμματισμού Ι

Συναρτήσεις

Εργίνα Καβαλλιεράτου (kavallieratou@aegean.gr)

Μόνιμη Επίκουρος Καθηγήτρια

Τμήμα Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

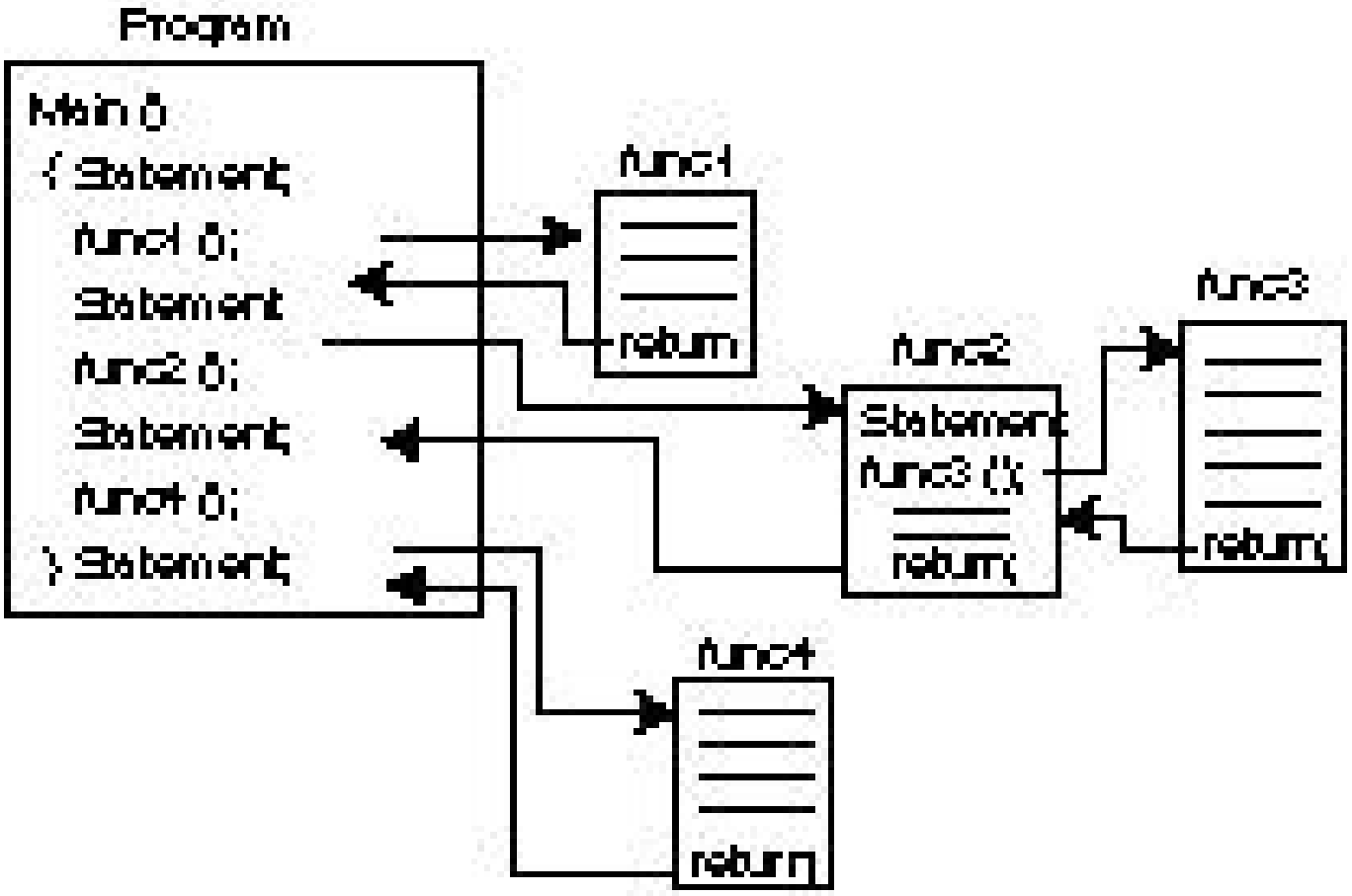
Σημερινό μάθημα

- ✓ C++ Συναρτήσεις
- ✓ Δήλωση συνάρτησης
- ✓ Σύνταξη συνάρτησης
- ✓ Πρότυπο συνάρτησης & συνάρτηση
- ✓ Αλληλο-καλούμενες συναρτήσεις
- ✓ Συναρτήσεις μαθηματικών
- ✓ Παράμετροι συναρτήσεων
- ✓ Τοπικές μεταβλητές - Block μεταβλητές
- ✓ Σφαιρικές μεταβλητές

C++ Συναρτήσεις

- ✓ Στον ΑΠ οι συναρτήσεις παραμένουν κεντρικό στοιχείο του προγράμματος.
- ✓ Η συνάρτηση είναι υποπρόγραμμα που επιδρά σε δεδομένα και μπορεί να επιστρέψει μια τιμή.
- ✓ Όλες οι συναρτήσεις στη C++ έχουν τύπο επιστροφής
  - Όταν δεν χρειάζεται να επιστρέψει τίποτα μια συνάρτηση παίρνει τον τύπο **`void`**.
- ✓ Επιπλέον δέχονται παραμέτρους για να δουλέψουν πάνω σε αυτές.
  - Κάθε παράμετρος έχει επίσης τύπο.

C++ Συναρτήσεις



C++ Συναρτήσεις

- ✓ Για να χρησιμοποιήσουμε μία συνάρτηση στο πρόγραμμα μας πρέπει πρώτα να τη δηλώσουμε και μετά να την ορίσουμε.
- ✓ Η δήλωση λέει στον compiler το όνομα της συνάρτησης, τι παραμέτρους παίρνει ως είσοδο.
- ✓ Ο ορισμός λέει στον compiler πως λειτουργεί μια συνάρτηση.
- ✓ Μία συνάρτηση δεν μπορεί να κληθεί από μία άλλη συνάρτηση αν δεν έχει πρώτα δηλωθεί.

Δήλωση συνάρτησης

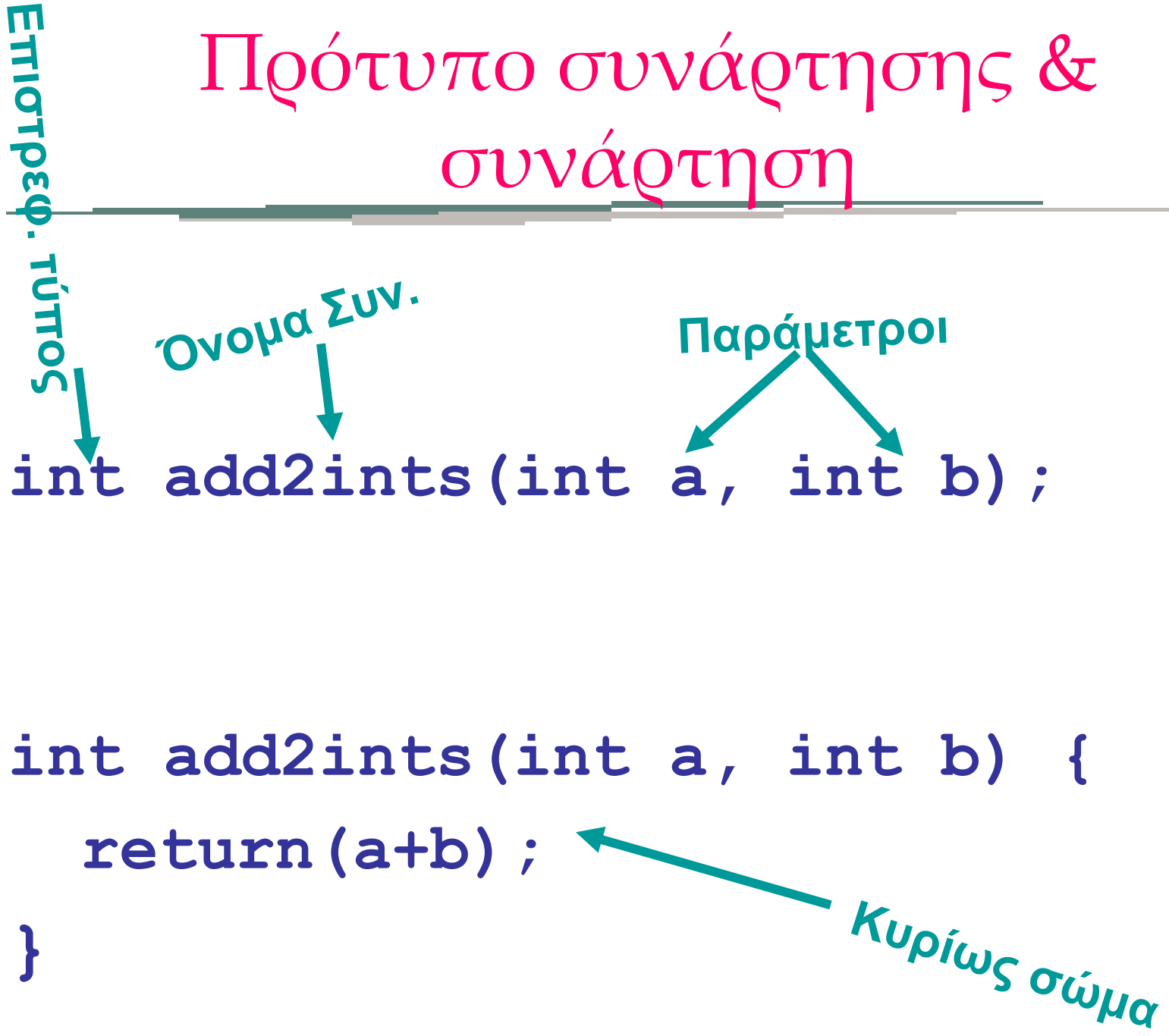
Υπάρχουν τρεις τρόποι να δηλώσουμε μία συνάρτηση:

1. Να γράψουμε το πρωτυπο της συνάρτησης σε ένα αρχείο και να δηλώσουμε με #include το αρχείο στο αρχείο του κυρίως προγράμματος.
2. Να γράψουμε το πρωτυπο της συνάρτησης στο αρχείο του προγράμματος πριν από τη χρήση της.
3. Να ορίσουμε τη συνάρτηση πριν από την κλήση της και έτσι δηλώνεται αυτόματα.

Σύνταξη συνάρτησης

- ✓ Πρέπει να αποφασίσουμε:
  - Τύπο επιστροφής
  - Όνομα συνάρτησης
  - Παράμετροι (τύπο & πλήθος)
- ✓ Να γράψουμε τον κώδικα του κυρίου σώματος

Πρότυπο συνάρτησης & συνάρτηση



Πρότυπο συνάρτησης & συνάρτηση

Σύνταξη Προτύπου

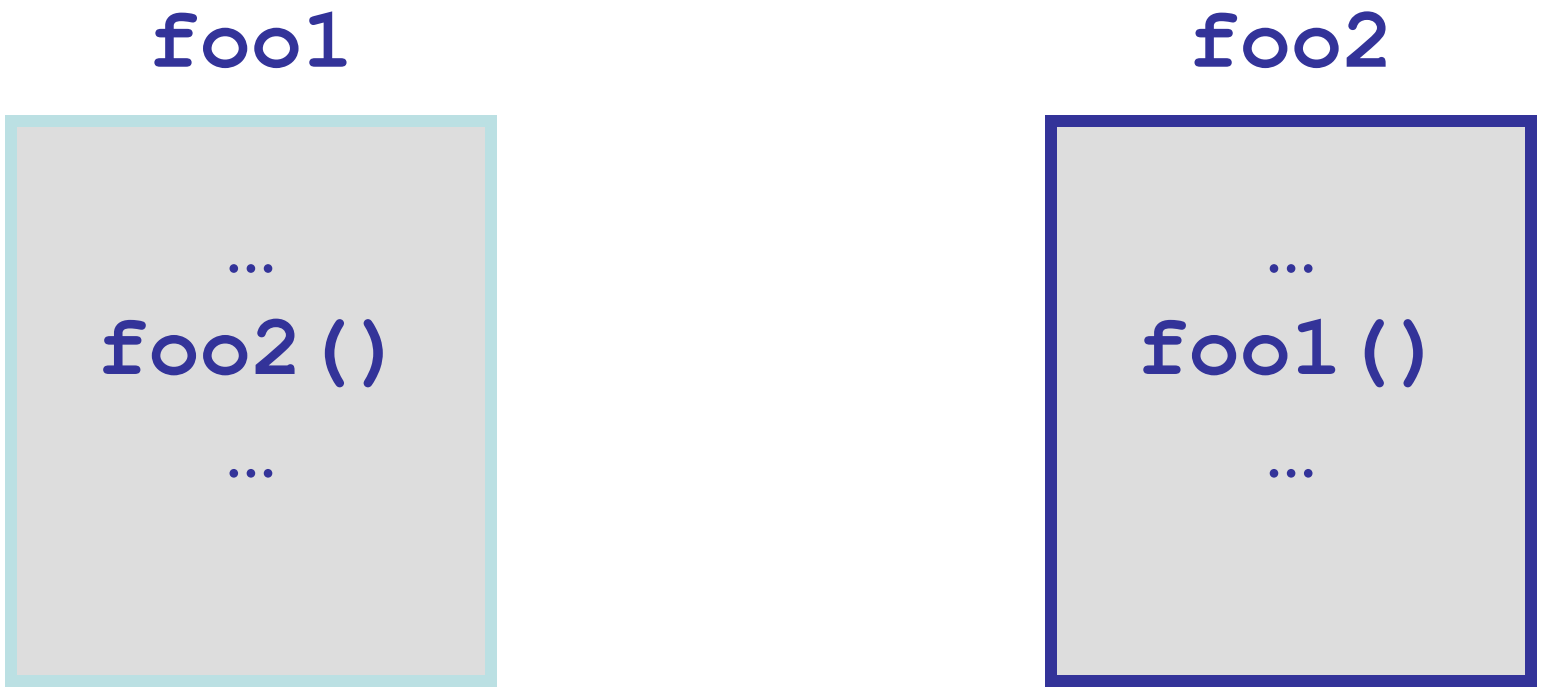
```
return_type function_name ( [type [parameterName]]...);
```

Σύνταξη συνάρτησης

```
return_type function_name ( [type parameterName]...)
{
   statements;
}
```

- ✓ Το πρότυπο συνάρτησης δε χρειάζεται να περιέχει τα ονόματα παραμέτρων αλλά επιβάλλεται να περιέχει τον τύπο τους.

Αλληλο-καλούμενες συναρτήσεις



Αλληλο-καλούμενες συναρτήσεις

```
char *chicken( int generation ) {
  if (generation == 0)
    return("Chicken!");
  else
    return(egg(generation-1));
}
```

```
char *egg( int generation ) {
  if (generation == 0)
    return("Egg!");
  else
    return(chicken(generation-1));
}
```

Αλληλο-καλούμενες συναρτήσεις

```
char *egg( int );
char *chicken( int );

int main(void) {
  int startnum;

  cout << "Enter starting generation of
  your chicken" << endl;
  cin >> startnum;
  cout << "Your chicken started as a " <<
            chicken(startnum) << endl;
  return(0);
  }
```

Παράδειγμα/1

```
typedef unsigned short USHORT;
#include <iostream.h>
USHORT FindArea(USHORT length, USHORT width);
int main()
{
  USHORT lengthOfYard;
  USHORT widthOfYard;
  USHORT areaOfYard;
  cout << "\nHow wide is your yard? ";
  cin >> widthOfYard;
  cout << "\nHow long is your yard? ";
  cin >> lengthOfYard;
```

Παράδειγμα/2

```
  areaOfYard= FindArea(lengthOfYard,widthOfYard);
  cout << "\nYour yard is ";
  cout << areaOfYard;
  cout << " square feet\n\n";
        return 0;
}

USHORT FindArea(USHORT l, USHORT w)
{
    return l * w;
}
```

double sqrt(double)

- ✓ Ανήκει στη βιβλιοθήκη math.h της C++
- ✓ Υπολογίζει τετραγωνική ρίζα
- ✓ Κατά την κλίση της **sqrt**, πρέπει να δώσουμε **double** ως παράμετρο.
- ✓ Επιστρέφει **double**.

```
x = sqrt(y);
x = sqrt(100);
```

`x = sqrt(y);`

- ✓ Το Υ είναι παράμετρος.
- ✓ Μια παράμετρος χρησιμοποιείται αλλά δεν αλλάζει.
- ✓ Αν το Υ είχε την τιμή 100 πριν την κλίση θα είναι 100 και μετά

Παράδειγμα

```
int i;
for (i=1;i<10;i++)
  cout << sqrt(i) << "\n";
```

- ✓ Η C++ *αλλάζει τον τύπο παραμέτρου αυτόματα*

compiler & `sqrt()`

- Πρέπει να πούμε στο compiler ότι θα χρησιμοποιηθεί η **`sqrt`**:

```
#include <math.h>
```

Άλλες συναρτήσεις μαθηματικών

```
cos     sin     tan

exp     log     log10   pow

fabs    fmod
```

Παράμετροι συναρτήσεων

- ✓ Οι παράμετροι λειτουργούν σαν τοπικές μεταβλητές μέσα στη συνάρτηση
  - Όταν κλιθεί η συνάρτηση θα περάσουν μέσα οι τιμές των παραμέτρων
  - Οι συναρτήσεις χρησιμοποιούν αντίγραφα των παραμέτρων για αυτό και οι αρχικές τιμές δεν αλλάζουν εκτός της συνάρτησης.

Παράδειγμα

```
int add2nums( int firstnum, int secondnum ) {
  int sum;

  sum = firstnum + secondnum;

  firstnum = 0;
  secondnum = 0;

  return(sum);
}
```

Τι γίνεται;

```
int add2nums(int a, int b) {
   a=a+b;
   return(a);
}
```

Testing `add2nums`

```
int main(void) {
  int y,a,b;

  cout << "Enter 2 numbers\n";
  cin >> a >> b;

  y = add2nums(a,b);

  cout << "a is " << a << endl;
  cout << "b is " << b << endl;
  cout << "y is " << y << endl;
  return(0);
}
```

Τοπικές μεταβλητές

- ✓ Οι παράμετροι και οι μεταβλητές που δηλώνονται μέσα σε μία συνάρτηση είναι **τοπικές**.
- ✓ Υπάρχουν μόνο μέσα στο σώμα της συνάρτησης
- ✓ Μετά τον τερματισμό της συνάρτησης παύουν να υπάρχουν

Block μεταβλητές

- Μπορούν επίσης να δηλωθούν μεταβλητές που θα υπάρξουν μόνο μέσα σε ένα ορισμένο block:

```
{
int foo;
      …
      …
}
```

Block μεταβλητές

```
void myFunc();
int main()    {
  int x = 5;
  cout << "\nIn main x is: " << x;
  myFunc();
  cout << "\nBack in main, x is: " << x;
  return 0;
}
```

Block μεταβλητές

```
void myFunc()   {
    int x = 8;
    cout << "\nIn myFunc, local x: " << x << endl;
    {
      cout << "\nIn block in myFunc, x is: " << x;
      int x = 9;
      cout << "\nVery local x: " << x;
    }
    cout << "\nOut of block, in myFunc, x: " << x << endl;
}
```

Block μεταβλητές - output

In main x is: 5
In myFunc, local x: 8

In block in myFunc, x is: 8
Very local x: 9
Out of block, in myFunc, x: 8

Back in main, x is: 5

Σφαιρικές μεταβλητές

- ✓ Μπορούμε να δηλώσουμε μεταβλητές έξω από κάθε συνάρτηση και ονομάζονται **σφαιρικές**
- ✓ Οι σφαιρικές μεταβλητές είναι προσβάσιμες και μπορούν να αλλαχθούν από όλες τις συναρτήσεις.

Τοπικές μεταβλητές– παράδειγμα/1

```
#include <iostream.h>
float Convert(float);
int main() {
   float TempFer;
   float TempCel;
  cout << "Please enter the temperature in Fahrenheit: ";
  cin >> TempFer;
  TempCel = Convert(TempFer);
   cout << "\nHere's the temperature in Celsius: ";
  cout << TempCel << endl;
        return 0;   }
```

Τοπικές μεταβλητές – παράδειγμα/2

```
float Convert(float TempFer) {
    float TempCel;
    TempCel = ((TempFer - 32) * 5) / 9;
    return TempCel; }
```

```
Please enter the temperature in Fahrenheit: 212

Here's the temperature in Celsius: 100
```

Σφαιρικές μεταβλητές

```
#include <iostream.h>
void myFunction();
int x = 5, y = 7;
int main()  {
    cout << "x from main: " << x << "\n";
    cout << "y from main: " << y << "\n\n";
    myFunction();
    cout << "Back from myFunction!\n\n";
    cout << "x from main: " << x << "\n";
    cout << "y from main: " << y << "\n";
    return 0;  }

void myFunction()  {
    int y = 10;
    cout << "x from myFunction: " << x << "\n";
    cout << "y from myFunction: " << y << "\n\n"; }
```

```
x from main: 5
y from main: 7

x from myFunction: 5
y from myFunction: 10

Back from myFunction!

x from main: 5
y from main: 7
```

Παράδειγμα 1.1

```
#include <iostream>
int addition (int a, int b){
  int r;
  r=a+b;
  return r;
}
void main (){
  int z;
  z = addition (5,3);
  cout << "The result is " << z;
}
```

Παράδειγμα 2.1

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

int subtraction (int a, int b){
  int r;
  r=a-b;
  return r;
}
```

Παράδειγμα 2.2

```
void main (){
  int x=5, y=3, z;
  z = subtraction (7,2);
  cout << "The first result is " << z << '\n';
  cout << "The second result is " << subtraction (7,2) << '\n';
  cout << "The third result is " << subtraction (x,y) << '\n';
  z= 4 + subtraction (x,y);
  cout << "The fourth result is " << z << '\n';
}
```

Παράδειγμα 3.1

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

void printmessage ()
{
  cout << "I'm a function!";
}
int main ()
{
  printmessage ();
}
```

Παράδειγμα 4.1

```
#include <iostream>
void duplicate (int& a, int& b, int& c){
  a*=2;
  b*=2;
  c*=2;
}
int main (){
  int x=1, y=3, z=7;
  duplicate (x, y, z);
  cout << "x=" << x << ", y=" << y << ", z=" << z;
  return 0;
}
```