Προγραμματισμός Υπολογιστών με C++

ΧΩΡΟΙ ΟΝΟΜΑΤΩΝ

Γεώργιος Παπαϊωάννου (2013-14)
gepap@aueb.gr

Περιεχόμενο Παρουσίασης

- Περιγραφή:
  - Τι είναι οι χώροι ονομάτων
  - Γιατί χρειαζόμαστε χώρους ονομάτων

- Τελευταία ενημέρωση: Ιούλιος 2013

Τι Είναι οι Χώροι Ονομάτων;

- Είναι ένας τρόπος οργάνωσης του κώδικα σε σύνολα με βάση τη λειτουργικότητα, την έκδοση ή οποιαδήποτε άλλη δόμηση επιθυμώ
- Συνήθως, η οργάνωση του κώδικα σε χώρους ονομάτων ακολουθεί την ομαδοποίηση κατά κοινή λειτουργικότητα ή σκοπό, σε βιβλιοθήκες κώδικα, αλλά δεν είναι υποχρεωτικό
- Ακολουθείται η ίδια τακτική με τα «πακέτα» της Java

Που μας Χρησιμεύουν οι Χώροι Ονομάτων;

- Στο να μη συγχέονται συναρτήσεις και κλάσεις με ίδιο όνομα αλλά από διαφορετικές βιβλιοθήκες
- Στο να έχουμε πολλαπλές εκδόσεις υλοποιήσεων για τις ίδιες λειτουργίες
- Στην εννοιολογική οργάνωση του κώδικά μας

Παράδειγμα Χρήσης Χώρου Ονομάτων

mylib.h

```
namespace auebsoft {
  float f1(int i);
  inline int f2(int i){
    return i * 10;
  }
}
```

mylib.cpp

```
#include "mylib.h"
namespace auebsoft {
  float f1(int i) {
    return i / 3.0F;
  }
}
```

myapp.cpp

```
#include <iostream>
using namespace std;
#include "mylib.h"

float f1(int i) {
   return 100 * i;
}

int main() {
 cout << f1(1)   // Τυπώνει 100.
    << auebsoft::f1(1) // 0.333333.
    << auebsoft::f2(1) // 10.
    << ::f1(1);     // Τυπώνει 100.
 cout << f2(1); // Λάθος: ανύπαρκτη
}               // συνάρτηση.
```

Η Εντολή using namespace

- Η εντολή **using namespace** μού επιτρέπει να αναφέρομαι στις συναρτήσεις, κλάσεις κλπ. ενός χώρου ονομάτων **χωρίς το «επώνυμό» τους**.
- Αν όμως υπάρχει συνάρτηση, κλάση κλπ. με το ίδιο όνομα και στον ανώνυμο χώρο («::»), τότε δημιουργείται **πρόβλημα αμφισημίας** και πρέπει να χρησιμοποιήσω οπωσδήποτε τον **τελεστή ::**.
- **Χωρίς το «using namespace std;»** θα έπρεπε να γράφω **std::string**, **std::cout**, **std::endl** κλπ.
- Αν θέλω να καλέσω μια συνάρτηση του «default» χώρου ονομάτων (δηλ. που δεν εντάσσεται σε χώρο ονομάτων), την καλώ με **::**ΟνομαΣυνάρτησης()

Η Εντολή using namespace - Παράδειγμα

mylib.h

```
namespace auebsoft {
  float f1(int i);
  inline int f2(int i){
    return i * 10;
  }
}
```

mylib.cpp

```
#include "mylib.h"
namespace auebsoft {
  float f1(int i) {
    return i / 3.0F;
  }
}
```

myapp.cpp

```
#include <iostream>
using namespace std;
#include "mylib.h"
using namespace auebsoft;
float f1(int i) {
   return 100 * i;
}

int main() {
 cout << f1(1)   // Τυπώνει 100.
    << auebsoft::f1(1) // 0.333333.
    << f2(1) // 10.
    << ::f1(1);     // Τυπώνει 100.
 cout << f1(1); // Λάθος: δεν ξέρει
                // ποια να διαλέξει
```

Παράδειγμα - STL Χωρίς using namespace

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    std::string s; // Η κλάση string ανήκει στο
                   // χώρο ονομάτων «std».
    // Το ίδιο και τα cout, cin, endl.
    std::cout << "Type a string: " << std::endl;
    std::cin >> s;
    std::cout << "You typed: " << s << std::endl;
}
```

Τι διαφορές Εντόπισα σε Σχέση με τη Java; (1)

Ομαδοποιεί κλάσεις, συναρτήσεις και άλλες δηλώσεις σε ονοματικούς χώρους (namespaces)	Ομαδοποιεί κλάσεις σε πακέτα (packages)
Οι χώροι ονομάτων δεν προϋποθέτουν αντιστοίχιση με δομή αρχείων στο δίσκο	Η ιεραρχία των πακέτων ακολουθεί αντίστοιχη ιεραρχία καταλόγων με .class αρχεία
Ο απροσδιόριστος namespace είναι ο κενός (default)	Το απροσδιόριστο πακέτο είναι το κενό (default)
Διαχωρίζω φωλιασμένες δηλώσεις namespaces και κλάσεων με ::	Διαχωρίζω φωλιασμένες δηλώσεις πακέτων και κλάσεων με .

Τι διαφορές Εντόπισα σε Σχέση με τη Java; (2)

C++

Μπορώ να παραλείψω το όνομα του namespace στη δήλωση μεταβλητής με τη using namespace ….

Εξακολουθώ να πρέπει να κάνω #include το κατάλληλο αρχείο δηλώσεων

Java

Μπορώ να παραλείψω το όνομα του πακέτου στη δήλωση μεταβλητών με το #import ….

Το #import δίνει και το μονοπάτι της κλάσης

Φωλιασμένοι Χώροι Ονομάτων

- Προφανώς, τόσο τις δηλώσεις χώρων ονομάτων όσο και τις αντίστοιχες κλήσεις σε περιεχόμενά τους μπορούμε να τα φωλιάσουμε:

```
namespace foo {
    namespace bar {
        void myFunction(int a);
    }
{
…
foo::bar::myFunction(10);
using namespace foo::bar;
myFunction(10);
```