ΤΕΙ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΑΝΟΙΚΤΑ ΑΚΑΔΗΜΑΙΚΑ ΜΑΘΗΜΑΤΑ

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ(Θ)

Ενότητα 4: ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

ΔΙΔΑΣΚΩΝ: ΠΑΡΙΣ ΜΑΣΤΟΡΟΚΩΣΤΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

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 ΤΕΙ Κεντρικής Μακεδονίας» 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.

Ενότητα 4

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

ΔΙΔΑΣΚΩΝ: ΠΑΡΙΣ ΜΑΣΤΟΡΟΚΩΣΤΑΣ

Περιεχόμενα ενότητας

1. Μετατροπή τύπων
2. Μετατροπές μεταξύ αντικειμένων και βασικών τύπων
3. Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

Σκοποί ενότητας

Μετατροπή τύπων

Γνωρίζουμε από πριν ότι προτάσεις όπως

```
x = y;
```

όπου x και y είναι π.χ. ακέραιου τύπου, αποδίδουν την τιμή μίας μεταβλητής σε μία άλλη.

Παρόμοια, η πρόταση

```
a1 = a2 + a3
```

όπου τα a1, a2, a3 είναι αντικείμενα, αποδίδει την τιμή ενός αντικειμένου σε ένα άλλο.

Έτσι, οι αποδόσεις τιμών μεταξύ μεταβλητών βασικών τύπων και τύπων ορισμένων από τον χρήστη, αναλαμβάνονται από το μεταγλωττιστή, αρκεί να είναι ο ίδιος τύπος δεδομένων και από τις δύο πλευρές του τελεστή ανάθεσης.

Όταν οι μεταβλητές είναι διαφορετικού τύπου, τότε, εάν είναι και οι δύο βασικού τύπου, η μετατροπή γίνεται αυτόματα από το μεταγλωττιστή (*casting*). Σε διαφορετική περίπτωση πρέπει εμείς να ορίσουμε τη λειτουργία, όπως θα αναλυθεί στις επόμενες διαφάνειες.

Μετατροπές μεταξύ αντικειμένων και βασικών τύπων

```
#include <cstdlib>
#include <iostream>

using namespace std;
const float MTF = 3.28033;
class EngDist
{
private:
        int feet;
        int  inches;
public:
        operator  float()       // συνάρτηση μετατροπής από τύπο οριζόμενο
        {                               // από τον χρήστη σε βασικό τύπο
                float  meters;
                meters = (feet + inches/12.0) / MTF;
                return meters;
        }
```

theory_4_casting_1.cpp

Μετατροπές μεταξύ αντικειμένων και βασικών τύπων

```
        EngDist();
        EngDist(int feet1, float  inches1);
        EngDist(float meters);
        void readDist();
        void printDist();
};      // τέλος της κλάσης

main()
{
        const int ft=6;
        const int in=10;
        EngDist  d1, d2(ft,in), d3(1.8295);
        float metr;
```

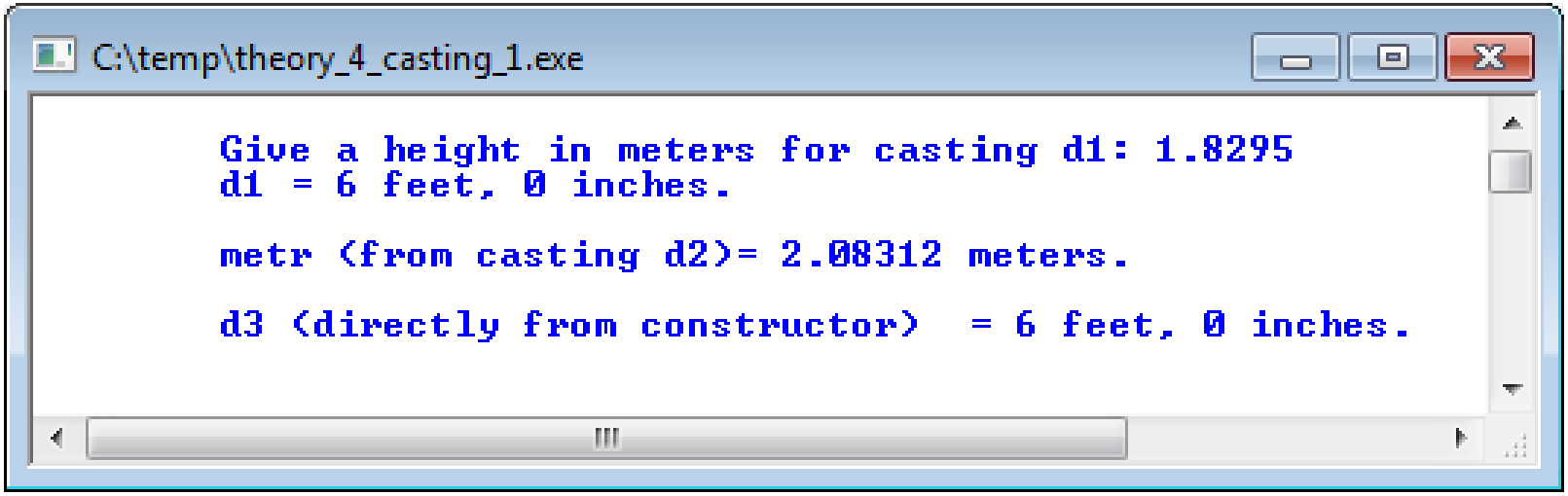
Εφαρμογή των 3 διαφορετικών δομητών

Μετατροπές μεταξύ αντικειμένων και βασικών τύπων

```
cout << endl << "\tGive a height in meters for casting d1: ";
cin >> metr;
d1 = metr;              // χρήση συνάρτησης δόμησης για μετατροπή από
cout << "\td1 = ";     // μέτρα σε EngDist
d1.printDist();

metr = d2;              // χρήση συνάρτησης μετατροπής για μετατροπή από
                        // EngDist σε μέτρα
cout << endl << "\tmetr (from casting d2)= " << metr << " meters." <<
                                                                  endl;
cout << endl << "\td3 (directly from constructor)  = "; // μέτρα σε
                                                        // EngDist
d3.printDist();
} // τέλος της main
```

Μετατροπές μεταξύ αντικειμένων και βασικών τύπων




```
Engdist :: EngDist(float meters)  // συνάρτηση δόμησης για μετατροπή
{                                 // από βασικό τύπο σε τύπο οριζόμενο από τον χρήστη
    float  ft;
    ft = MTF * meters;
    feet = int(ft);
    inches = 12 * (ft - feet);
```

Μετατροπές μεταξύ αντικειμένων και βασικών τύπων

```
EngDist :: EngDist()
{
  feet = 0;
  inches = 0;
}
EngDist :: EngDist(int feet1, int  inches1)
{
  feet = feet1;
  inches = inches1;
}
```

Μετατροπές μεταξύ αντικειμένων και βασικών τύπων

```
void EngDist :: readDist()
{
        cout << endl << "\tGive feet:";
        cin >> feet;
        cout << endl << "\tGive inches:";
        cin >> inches;
}
void EngDist :: printDist()
{
        cout << feet << " feet, " << inches << " inches." << endl;
}
```

Μετατροπές μεταξύ αντικειμένων και βασικών τύπων

Συνοψίζοντας, για να μετατρέψουμε ένα βασικό τύπο fundamental_type - όπως π.χ. float- σε τύπο οριζόμενο από τον χρήστη - π.χ. EngDist- χρησιμοποιούμε μία συνάρτηση δόμησης με όρισμα του βασικού τύπου fundamental_type: **EngDist(float meters)**.

Αυτή η συνάρτηση εκτελείται όταν εκτελείται η πρόταση:

d1 = 1.95;

Όταν μετατρέπουμε ένα τύπο οριζόμενο από το χρήστη σε βασικό, χρησιμοποιούμε τη συνάρτηση μετατροπής: **operator float()**.

Η συνάρτηση καλείται όταν εκτελείται η πρόταση: **metr = d2;**

Μόλις ο μεταγλωττιστής αντιληφθεί ότι προσπαθούμε να μετατρέψουμε έναν τύπο οριζόμενο από το χρήστη σε ένα βασικό τύπο, αναζητά έναν τελεστή με υπερφόρτωση **=**. Όταν δε βρει κάποιον συνεχίζει την αναζήτηση, βρίσκει τη συνάρτηση μετατροπής και τη χρησιμοποιεί.

Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

- Όταν έχουμε να εκτελέσουμε μετατροπή μεταξύ δύο αντικειμένων που προέρχονται από διαφορετικές κλάσεις, ακολουθούμε μία παρόμοια διαδικασία. Χρησιμοποιούμε μία συνάρτηση δόμησης με ένα όρισμα, αν θέλουμε η ρουτίνα μετατροπής να βρίσκεται στην κλάση του αντικειμένου προορισμού (δηλαδή αριστερά από το =), ενώ χρησιμοποιούμε μία συνάρτηση μετατροπής, αν θέλουμε η ρουτίνα μετατροπής να βρίσκεται στην κλάση του αντικειμένου προέλευσης (δηλαδή δεξιά από το =).

- Τα δύο επόμενα παραδείγματα υλοποιούν τις δύο αυτές περιπτώσεις, για τη μετατροπή μίας απόστασης εκφρασμένης στο αγγλοσαξωνικό σύστημα μέτρησης (πόδια και ίντσες) σε μία απόσταση εκφρασμένης στο διεθνές σύστημα μέτρησης (μέτρα και εκατοστά).

Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

1) Ρουτίνα στο αντικείμενο προορισμού

```
const float MTF = 3.28033;
class EngDist
{
private:
    int feet;
    int  inches;
public:
    EngDist()
    {
        feet = 0;
        inches = 0;
    }
    EngDist(int feet1, int  inches1)
    {
        feet = feet1;
        inches = inches1;
    }
```

theory_4_casting_2.cpp

Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

```
    void printDist()
    {
          cout << feet << " feet, " << inches << " inches." << endl;
    }

    int  getFeet()
    {
          return  feet;
    }

    float  getInches()
    {
          return  inches;
    }
};  // τέλος της κλάσης EngDist
```

Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

```
class SI_Dist
{
private:
    int m;
    int  cm;
public:
    SI_Dist()
    {
        m = 0;
        cm = 0;
    }
    SI_Dist(int m1, int  cm1)
    {
        m = m1;
        cm = cm1;
    }
```

Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

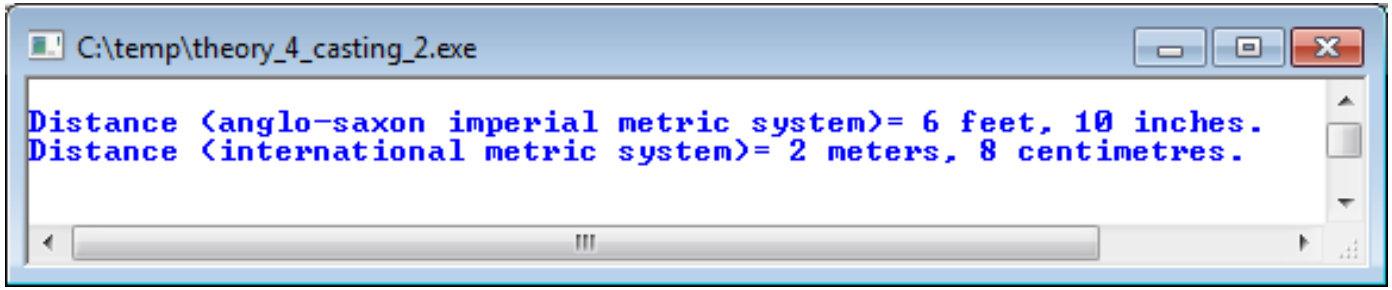
```
SI_Dist(EngDist e)
 {
        int  ft, in;
        float  mf;
        ft = e.getFeet();
        in = e.getInches();
        mf = (ft + in/12.0) / MTF;
        m = int(mf);
        cm = int((mf - m) * 100);
 }
 void printDist()
 {
        cout << m << " meters, " << cm << " centimetres." << endl;
 }
}; // τέλος της κλάσης SI_Dist
```

Για να δοθούν εκατοστά χωρίς δεκα-δικά.

Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

```
main()
{
    SI_Dist  gr;
   EngDist  eng(6, 10.0);

   gr = eng;
   cout << “Distance (anglo-saxon imperial metric system)= “;
   eng.printDist();
   cout << “Distance (international metric system)= “;
   gr.printDist();
}
```



Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

2) Ρουτίνα στο αντικείμενο προέλευσης

```
const float MTF = 3.28033;
class SI_Dist
{
private:
    int m;          int  cm;
public:
    SI_Dist()
    {
        m = 0;  cm = 0;
    }
    SI_Dist(int m1, int  cm1)
    {
        m = m1;          cm = cm1;
    }
```

theory_4_casting_3.cpp

Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

```
    void printDist()
    {
        cout << m << " meters, " << cm << " centimetres." << endl;
    }
};  // τέλος της κλάσης SI_Dist
```

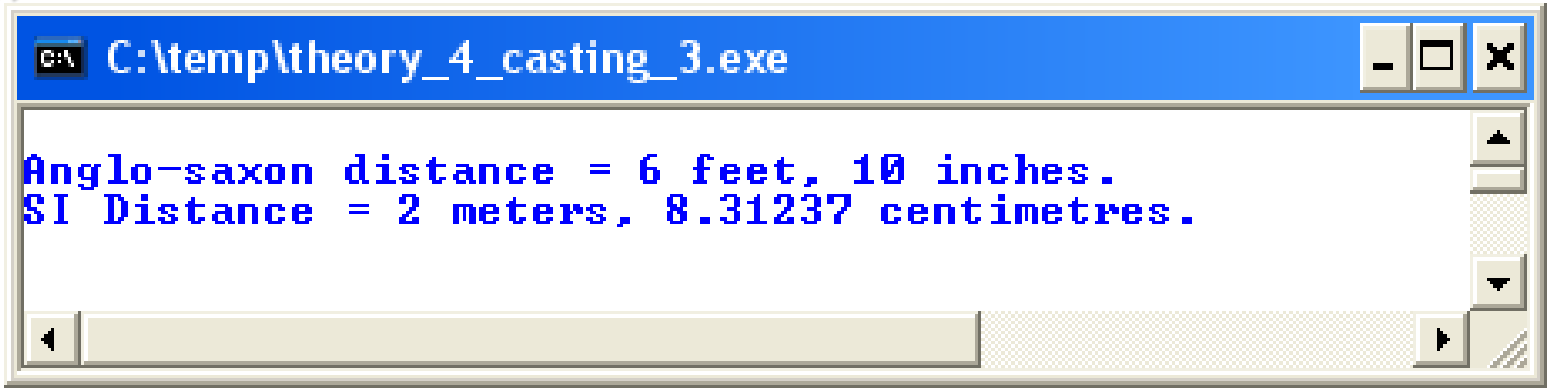
```
class EngDist
{
private:
    int feet;
    int  inches;
public:
    EngDist()
    {
        feet = 0;          inches = 0;
    }
```

Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

```
    EngDist(int feet1, int  inches1)
    {
          feet = feet1;      inches = inches1;
    }
    void printDist()
    {
          cout << feet << " feet, " << inches << " inches." << endl;
    }
    operator SI_Dist()
    {
          int met, ekat;     float mf;
          mf = (feet+inches/12.0) / MTF;
          met = int(mf);
          ekat = int((mf-met)*100);
          return  SI_Dist(met, ekat);
    }
};   // τέλος της κλάσης EngDist
```

Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

```
main()
{
    SI_Dist  gr;
    EngDist  eng(6, 10.0);
    gr = eng;
    cout << "Anglo-saxon distance = ";
    eng.printDist();
    cout << "SI Distance = ";
    gr.printDist();
}
```



Τέλος Ενότητας