ΤΕΙ ΣΕΡΡΩΝ

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Υπερφόρτωση Τελεστών
Μετατροπή Τύπων

Ευάγγελος Γ. Ούτσιος
Θεόδωρος Γ. Λάντζος

Διάλεξη Νο5

Εξέλιξη Μαθήματος

- Ο δρόμος για ΟΟ
- Η σκοπός του ΟΟ
- Δομή ΟΟ προγράμματος και κλάση
- Ορισμός και διαχείριση κλάσεων
- Συναρτήσεις εγκατάστασης
- Υπερφόρτωση συναρτήσεων εγκατάστασης
- Αντικείμενα ως ορίσματα συναρτήσεων
- Αντικείμενα ως επιστρεφόμενες τιμές συναρτήσεων
- Πίνακες μονοδιάστατοι & πολλών διαστάσεων
- Πίνακες σαν δεδομένα κλάσεων
- Πίνακες αντικειμένων

Υπερφόρτωση τελεστών

- Όπως υπερφορτώνουμε συναρτήσεις μπορούμε να υπερφορτώσουμε και τελέστές.
- Η δυνατότητα χρήσης τελεστών σε τύπους δεδομένων ορισμένων από το χρήστη
- Επίσης ως η χρήση των ίδιων τελεστών για διαφορετικό σκοπό

Υπερφόρτωση Αριθμητικών Τελεστών

```
#include <iostream.h>
class Account
{
   private:
        float  balance;
   public:
        Account()
        {
           balance = 0;
        }
        Account(float  balance1)
        {
           balance = balance1;
        }
        void  withdraw(float  money)
        {
           if (money <= balance)
              balance = balance – money;
           else
              cout << "Το ποσό ανάληψης υπερβαίνει το τρέχον!" << endl;
        }
        void deposit(float  money)
        {
           balance += money;
        }
        float getBalance()
        {
           return  balance;
        }
        Account  operator + (Account  ac)
        {
           Account  temp;
           temp.balance = balance + ac.balance;
           return temp;
        }
};
main()
{
   Account  ac1(100.0), ac2(70.0), ac3;
   ac3 = ac1 + ac2;
   cout << "Τρέχον ποσό λογαριασμού ac1:" << ac1.getBalance() << endl;
   cout << "Τρέχον ποσό λογαριασμού ac2:" << ac2.getBalance() << endl;
   cout << "Συνολικό ποσό λογαριασμών:" << ac3.getBalance() << endl;
}
```

Δεσμευμένη λέξη operator

Χρήση

Ένα όρισμα λιγότερο από τον αριθμό των τελεστέων, αφού ο ένας τελεστέος είναι το αντικείμενο του οποίου μέλος είναι η συνάρτηση του τελεστή

Υπερφόρτωση τελεστών σύγκρισης

```
#include <iostream.h>
class Account
{
  private:
      float  balance;
  public:
      Account()
      {
         balance = 0;
      }
      Account(float  balance1)
      {
         balance = balance1;
      }
      void  withdraw(float  money)
      {
         if (money <= balance)
            balance = balance – money;
         else
            cout << "Το ποσό ανάληψης υπερβαίνει το τρέχον!" << endl;
      }
      void deposit(float  money)
      {
         balance += money;
      }
      float getBalance()
      {
         return  balance;
      }
      bool  operator > (Account  ac)
      {
         if (balance > ac.balance)
               return true;
         else
               return false;
      }
};
```

Ορισμός

Κλήση

```
main()
{
   Account  ac1(100.0), ac2(70.0);

   if (ac1 > ac2)
      cout << "Το ποσό του λογαριασμού ac1 είναι μεγαλύτερο." << endl;
   else
      cout << "Το ποσό του ac2 είναι μεγαλύτερο ή είναι ίσοι." << endl;
}
```

Μετατροπές τύπων

- Όταν έχουμε τελεστέους διαφορετικών τύπων αυτοί μετατρέπονται σε ενιαίο
  - (α) Με αυτόματο τρόπο (Υπονοούμενη)
  - (β) Με ρητή εντολή του προγραμματιστή
- Α. Υπονοούμενη
  - 3.0+ ½- >3.0 και όχι 3.5
  - Όταν έχουμε δύο τύπους δεδομένων, ο στενότερος μετατρέπεται στον ευρύτερο χωρίς απώλεια
  - Οι τύποι της γλώσσας ταξινομούνται ανάλογα με το μέγεθος μνήμης που απαιτούν για αποθήκευση
    - Char < int < long < float < double
  - Στην γλώσσα C οι αριθμητικές εκφράσεις μετατρέπουν αυτόματα τον τύπο char σε int και το float σε double

Β. Ρητές μετατροπές

- Μετατροπή μιας τιμής σε διαφορετικό τύπο κατόπιν εντολής του προγραμματιστή
- Η διαδικασία ονομάζεται προσαρμογή
- Ο τελεστής μετατροπής είναι μοναδιαίος και έχει την μορφή (τύπος δεδομένων)
- Τοποθετείται μπροστά από μια έκφραση για να μετατρέψει την τιμή της στον περικλειόμενο σε παρενθέσεις τύπο
  - 5/2 -> 2
  - (float) 5 / (float) 2 -> 2.5

Μετατροπές μεταξύ αντικειμένων και βασικών τύπων

```
#include <iostream.h>
const float MTF = 3.28033;
class EngDist
{
  private:
      int feet;
      float  inches;
  public:
      EngDist()
      {
        feet = 0;
        inches = 0;
      }
      EngDist(float meters)   // συνάρτηση εγκατάστασης για μετατροπή από
      {                       // βασικό τύπο, σε τύπο ορισμένο από το χρήστη
        float  ft;
        ft = MTF * meters;
        feet = int(ft);
        inches = 12 * (ft - feet);
      }
      EngDist(int feet1, float  inches1)
      {
        feet = feet1;
        inches = inches1;
      }
      void  readDist()
      {
        cout << "Give feet:";
        cin >> feet;
        cout << "Give inches:";
        cin >> inches;
      }
      void printDist()
      {
        cout << feet << " feet, " << inches << " inches." << endl;
      }
      operator  float()
      {
        float  meters;
        meters = (feet + inches/12) / MTF;
        return meters;
      }
};
main()
{
  EngDist  d1, d2(5, 10.0);
  float metr;

  d1 = 1.95;          // χρήση συνάρτησης εγκατάστασης για μετατροπή από
  cout << "d1 = ";    // μέτρα σε EngDist
  d1.printDist() ;

  metr = d2 ;         // χρήση συνάρτησης μετατροπής  για μετατροπή από
                      // EngDist σε μέτρα
  cout << "d2 = " << metr << " μέτρα." << endl;
}
```

Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

1) Ρουτίνα στο αντικείμενο προορισμού

```
#include <iostream.h>
const float MTF = 3.28033;
class EngDist
{
  private:
      int feet;
      float  inches;
  public:
      EngDist()
      {
         feet = 0;
         inches = 0;
      }
      EngDist(int feet1, float  inches1)
      {
         feet = feet1;
         inches = inches1;
      }
      void printDist()
      {
         cout << feet << " feet, " << inches << " inches." << endl;
      }
      int  getFeet()
      {
         return  feet;
      }
      float  getInches()
      {
         return  inches;
      }
};

class GrDist
{
  private:
      int m;
      float  cm;
  public:
      GrDist()
      {
         m = 0;
         cm = 0;
      }
      GrDist(int m1, float  cm1)
      {
         m = m1;
         cm = cm1;
      }
      GrDist(EngDist e)
      {
         int  ft;
         float  in, mf;
         ft = e.getFeet();
         in = e.getInches();
         mf = (ft + in/12) / MTF;
         m = int(mf);
         cm = (mf – m) * 100;
      }
      void printDist()
      {
         cout << m << " meters, " << cm << " centimetres." << endl;
      }
};
```

Αντικείμενο προορισμού

Ρουτίνα

```
main()
{
   GrDist  gr;
   EngDist  eng(5, 10.0);

   gr = eng;
   cout << "English distance = ";
   eng.printDist();

   cout << "Greek Distance = ";
   gr.printDist();
}
```

Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

```
#include <iostream.h>
const float MTF = 3.28033;

class GrDist
{
   private:
        int m;
        float  cm;
   public:
        GrDist()
        {
           m = 0;
           cm = 0;
        }
        GrDist(int m1, float  cm1)
        {
           m = m1;
           cm = cm1;
        }
        void printDist()
        {
           cout << m << " meters, " << cm << " centimetres." << endl;
        }
};
class EngDist
{
   private:
        int feet;
        float  inches;
   public:
        EngDist()
        {
           feet = 0;
           inches = 0;
        }
        EngDist(int feet1, float  inches1)
        {
           feet = feet1;
           inches = inches1;
        }
        void printDist()
        {
           cout << feet << " feet, " << inches << " inches." << endl;
        }
        operator GrDist()
        {
           int met;
           float ekat, mf;
           mf = (feet+inches/12)/MTF;
           met = int(mf);
           ekat = (mf-met)*100;
           return  GrDist(met, ekat);
        }
};
main()
{
   GrDist  gr;
   EngDist  eng(5, 10.0);

   gr = eng;
   cout << "English distance = ";
   eng.printDist();

   cout << "Greek Distance = ";
   gr.printDist();
}
```

Αντικείμενο προέλευσης