Υπερφόρτωση Τελεστών

Υπερφόρτωση Τελεστών

- ✓ Από τα πιο ενδιαφέροντα χαρακτηριστικά του Αντικειμενοστραφούς Προγραμματισμού.
- ✓ Αναφέρεται στη πρόσθετη χρήση των συνηθισμένων τελεστών, όπως π.χ. +,-,< κ.α., σε τύπους δεδομένων οριζόμενους από τον χρήστη.
- ✓ Ως τώρα έχουμε δει να χρησιμοποιούμε αρ. τελεστές μόνο σε πράξεις. Με την υπερφόρτωση τελεστών θα μάθουμε να εκτελούμε πράξεις μεταξύ αντικειμένων μιας κλάσης.

Υπερφόρτωση αρ. Τελεστών

- ✓ Στο παράδειγμα άθροισης των λογαριασμών της διάλεξης 5 παρουσιάστηκε τρόπος με τον οποίο δύο αντικείμενα της κλάσης Account μπορούσαν να αθροιστούν.

```
ac3 = ac1.addBalance(ac2);
```

- ✓ Εάν χρησιμοποιηθεί ο τελεστής + με υπερφόρτωση, τότε θα μπορούσαμε να είχαμε την εξής πρόταση:

```
ac3 = ac1 + ac2;
```

- ✓ Το πλήρες πρόγραμμα έχει ως εξής:

```
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;
 }
```

Υπερφόρτωση αρ. Τελεστών

Για να «αναγκάσουμε» τον τελεστή + να ενεργήσει πάνω σε ένα αντικείμενο, χρησιμοποιούμε τη δεσμευμένη λέξη operator. Συγκεκριμένα, γράφουμε μία συνάρτηση *όπου ο τύπος επιστρεφόμενης τιμής μπαίνει πρώτος, ακολουθούμενος από τη δεσμευμένη λέξη operator και αμέσως μετά τον ίδιο τον τελεστή*. Τέλος, μέσα σε παρενθέσεις γράφουμε τη λίστα των ορισμάτων.

Όταν στην καλούσα συνάρτηση εκτελείται η πρόταση:

```
ac3 = ac1 + ac2;
```

τότε γίνεται υπερφόρτωση του τελεστή + (επειδή τα ac1 και ac2 έχουν ορισθεί ως αντικείμενα), προστίθενται τα αντικείμενα ac1 και ac2 και το αποτέλεσμα αποδίδεται στο αντικείμενο ac3.

Θα πρέπει να διευκρινισθεί ότι η συνάρτηση χρησιμοποιεί ως όρισμα το αντικείμενο που βρίσκεται δεξιά του τελεστή (π.χ. το ac2). Ακόμα, η *συνάρτηση είναι μέλος στο αντικείμενο που βρίσκεται αριστερά του τελεστή* (π.χ. στο ac1) και έτσι η αναφορά στα δεδομένα αυτού του αντικειμένου είναι άμεση.

Υπερφόρτωση Τελεστών Σύγκρισης

- ✓ Με παρόμοιο τρόπο υπερφορτώνουμε τελεστές σύγκρισης.

- ✓ Στο επόμενο πρόγραμμα θα χρησιμοποιήσουμε τον τελεστή > (μεγαλύτερο από), με υπερφόρτωση στη κλάση Account, για να μπορέσουμε να συγκρίνουμε τους δύο λογαριασμούς.

```
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;
    }
};    // τέλος της κλάσης
```

```
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;
}
```

Τελεστές που δεν επιδέχονται υπερφόρτωση

- ✓ Οι τελεστές:
  - :: (Διάκρισης εμβέλειας)
  - . (Πρόσβαση σε μέλος)
  - .* (Πρόσβαση σε μέλος μέσω δείκτη σε μέλος)
  - ?: (Υποθετικός τελεστής)

- ✓ Δεν μπορούν να δημιουργηθούν νέοι τελεστές της μορφής ** ή κ.α.

Υπερφόρτωση τελεστών με συναρτήσεις που δεν είναι μέλη κλάσης

- Οι συναρτήσεις υπερφόρτωσης των τελεστών δεν είναι απαραίτητο να είναι μέλη κάποιας κλάσης.
- Στην περίπτωση αυτή, δέχονται δύο ορίσματα.
  - Το πρώτο όρισμα είναι το αντικείμενο που γράφεται αριστερά από τον τελεστή και το δεύτερο εκείνο που γράφεται δεξιά του.
  - Στα προηγούμενα παραδείγματα, οι συναρτήσεις υπερφόρτωσης των τελεστών + και > θα μπορούσαν να οριστούν έξω από την κλάση ως

    Account operator +(Account ac1, Account ac2) και

    bool operator >(Account ac1, Account ac2)
- Εάν οι συναρτήσεις αυτές χρησιμοποιούν ιδιωτικά δεδομένα της κλάσης Account, θα πρέπει να δηλωθούν και μέσα στην κλάση ως φιλικές (friend).

  friend Account operator +(Account, Account);

```
#include <iostream>

using namespace std;

class Rectangle {
  private:
    float side_a, side_b;
  public:
    Rectangle(float a,float b) {
      side_a = a;
      side_b = b;
    }
    float area() {
      return side_a*side_b;
    }
};

bool operator>(Rectangle rec1, Rectangle rec2) {
   return (rec1.area() > rec2.area());
}
```

```
int main()
{
   float a,b;
   cout << "Enter dimensions of first rectangle\n";
   cout << "length: ";
   cin >> a;
   cout << "width: ";
   cin >> b;
   Rectangle r1(a,b);
   cout << "Enter dimensions of second rectangle\n";
   cout << "length: ";
   cin >> a;
   cout << "width: ";
   cin >> b;
   Rectangle r2(a,b);
   if (r1 > r2) {
      cout << "The first rectangle is bigger.\n";
   } else if (r2 > r1) {
      cout << "The second rectangle is bigger.\n";
   } else
      cout << "The two rectangles have the same area.\n";
   return 0;
}
```

Υπερφόρτωση τελεστών ++ και --

- Όπως γνωρίζουμε, οι τελεστές αυτοί μπορούν να χρησιμοποιηθούν είτε σε προθεματική (prefix) είτε σε μεταθεματική (postfix) μορφή.
- Οι δύο αυτές μορφές χρειάζονται να υπερφορτωθούν ξεχωριστά.
- Για να υπερφορτώσουμε την προθεματική μορφή με ορίζουμε μια συνάρτηση-μέλος χωρίς παραμέτρους.
- Για να υπερφορτώσουμε την μεταθεματική μορφή, ορίζουμε μια συνάρτηση-μέλος με μια ακέραια παράμετρο. Κατά τη χρήση του τελεστή, η παράμετρος αυτή παίρνει αυτόματα την τιμή 0.
- Εάν θέλουμε να χρησιμοποιήσουμε συναρτήσεις που δεν είναι μέλη της κλάσης, θα πρέπει να προσθέσουμε μια πρώτη παράμετρο, η οποία θα είναι το αντικείμενο στο οποίο εφαρμόζουμε τον τελεστή.

Το παρακάτω πρόγραμμα δείχνει την κλάση Rectangle του προηγούμενου παραδείγματος, στην οποία έχουν προστεθεί συναρτήσεις-μέλη που υπερφορτώνουν τον τελεστή ++ και στην προθεματική και στην μεταθεματική του μορφή. Επιπλέον έχει προστεθεί μια συνάρτηση-μέλος show που εμφανίζει στην οθόνη τις διατάσεις του ορθογωνίου.

```
#include <iostream>
#include <string>

using namespace std;

class Rectangle {
  private:
    float side_a, side_b;
  public:
    Rectangle() {
       side_a = 0.0;
       side_b = 0.0;
    }
    Rectangle(float a,float b) {
       side_a = a;
       side_b = b;
    }
```

Η κλάση Rectangle (συνέχεια από την προηγούμενη διαφάνεια)

```
    void show(string name) {
        cout << "Rectangle " << name << " is " << side_a << "x" << side_b <<endl;
    }
    Rectangle operator ++() { //overload prefix form
        side_a ++;
        side_b ++;
        return *this;
    }
    Rectangle operator ++(int i) { //overload postfix form
         Rectangle r = *this;
         side_a ++;
         side_b ++;
         return r;
    }
}; //end class
```

Η μέθοδος main

```
int main()
{
   Rectangle r(2.0,1.0);
   Rectangle r1;
   r.show("r");
   r1.show("r1");
   r1=++r;
   r.show("r");
   r1.show("r1");
   r1=r++;
   r.show("r");
   r1.show("r1");
   return 0;
}
```

Αποτελέσματα στην οθόνη

```
Rectangle r is 2x1
Rectangle r1 is 0x0
Rectangle r is 3x2
Rectangle r1 is 3x2
Rectangle r is 4x3
Rectangle r1 is 3x2
```

Αν θέλαμε οι συναρτήσεις υπερφόρτωσης του τελεστή να μην ήταν μέλη της κλάσης θα τις ορίζαμε έξω από την κλάση, ως εξής

```
Rectangle operator ++(Rectangle &r) {
    cout << "pre\n";
    r.side_a ++;
    r.side_b ++;
    cout << r.side_a;
    return r;
}
Rectangle operator ++(Rectangle &r,int x) {
    Rectangle r1 = r;
    cout << "post\n";
    r.side_a ++;
    r.side_b ++;
    return r1;
}
```

Η παράμετρος r παριστάνει το αντικείμενο στο οποίο εφαρμόζεται ο τελεστής. Δηλώνεται με αναφορά διότι ο τελεστής θέλουμε να αλλάζει το αντικείμενο και όχι απλώς να χρησιμοποιεί την τιμή του.

Μέσα στον κώδικα της κλάσης πρέπει να υπάρχουν οι δηλώσεις

```
friend Rectangle operator ++(Rectangle &);
friend Rectangle operator ++(Rectangle &,int);
```

Αν προσθέσουμε στο τροποποιημένο αυτό πρόγραμμα την ίδια συνάρτηση main όπως πριν και την εκτελέσουμε, θα πάρουμε τα ίδια ακριβώς αποτελέσματα.

Τρεις υπερφορτωμένοι τελεστές που παρέχει η C++ και τους χρησιμοποιούμε συνέχεια χωρίς να το καταλαβαίνουμε είναι ο = (όταν χρησιμοποιείται με αντικείμενα), και οι << και >> που βάζουμε στις εντολές εισόδου/εξόδου.