Υπερφόρτωση τελεστών

#6

Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων (Άρτα)
Γκόγκος Χρήστος

V 1.1

Τελεστές

- Υπάρχουν πολλοί διαθέσιμοι τελεστές (+, -, *, /, ==, !=, <<, >>) που λειτουργούν με τους ενσωματωμένους τύπους (π.χ. `int, double`).
- Μπορούμε να φανταστούμε ότι οι τελεστές αυτοί υλοποιούνται με κλήσεις συναρτήσεων που γίνονται αυτόματα από τη C++:
  - Για παράδειγμα, η εντολή:

```
int x = 1 + 2;
```

  - Μπορεί να αντικατασταθεί αυτόματα από μια κλήση συνάρτησης όπως η ακόλουθη:

```
int x = operator+(1,2);
```

  - Με το ακόλουθο πρωτότυπο:

```
int operator+(int p1, int p2); /* επιστρέφει το άθροισμα των
                                  p1 και p2 */
```

Τελεστές

- Ένα ακόμα παράδειγμα:

```
double var = 1.1;
double x = 4.3 * 2.1 - var;
```

- Θα μπορούσε να αντικατασταθεί αυτόματα από τις εξής κλήσεις συναρτήσεων:

```
double x = operator-(operator*(4.3, 2.1), var);
```

- Με τα ακόλουθα πρωτότυπα:

```
double operator*(double p1, double p2);
double operator-(double p1, double p2);
```

- Η C++ υποστηρίζει την υπερφόρτωση τελεστών που επιτρέπει σε γνώριμους τελεστές, όπως το +, να χρησιμοποιούνται με τύπους ορισμένους από τον χρήστη (όπως είναι τα αντικείμενα).

Παράδειγμα κίνητρο

- Παράδειγμα 1
  - Παρατηρήστε τη χρήση των αριθμητικών τελεστών στο ακόλουθο παράδειγμα:

```
int x = 3, y = 6, z;
float a=3.4, b = 2.1, c;
z = x + y;
c = a / b;
```

  - Έστω ότι επιθυμούμε να πραγματοποιούμε πράξεις με αντικείμενα Fraction με τον ίδιο γνώριμο τρόπο:

```
1/2 + 3/4;
2/3 - 1/3;
```

  - Θα μπορούσαμε να το πετύχουμε αυτό χρησιμοποιώντας την κλάση Fraction ως εξής;

```
Fraction f1(1,2), f2(3,4), f3;
f3 = f1 + f2; /* θα το δεχθεί ο μεταγλωττιστής; */
```

    - Θα πρέπει να είναι σαφές ότι το παραπάνω δεν έχει νόημα για τον μεταγλωττιστή απευθείας. Το Fraction είναι ένας τύπος που έχει οριστεί από τον χρήστη. Πώς θα μπορούσε να γνωρίζει ο υπολογιστής περί κοινών παρονομαστών κ.λπ.;
    - Ωστόσο, θα ήταν καλό να μπορούσαμε να χρησιμοποιούμε τα αντικείμενα Fraction με τον ίδιο τρόπο με τον οποίο χρησιμοποιούμε και τους άλλους αριθμητικούς τύπους (όπως το int και το double).

Παράδειγμα κίνητρο

- Παράδειγμα 2:
  - Θεωρείστε την έξοδο του ακόλουθου κώδικα που γνωρίζουμε ότι είναι έγκυρη:

```
int x = 5;
cout << x;
```

  - Τι θα συμβεί όμως στην ακόλουθη περίπτωση;

```
Fraction f(3,4);
cout << “The fraction f = ” << f << endl;
```

    - Πάλι, θα πρέπει να είναι σαφές ότι ο μεταγλωττιστής δεν θα μπορούσε να γνωρίζει πως το αντικείμενο Fraction θα έπρεπε να εμφανίζεται στην οθόνη.

  - Μπορούμε να χρησιμοποιούμε υπερφόρτωση τελεστών για να παρέχουμε μια συνάρτηση που υλοποιεί την επιθυμητή συμπεριφορά για μια λειτουργία όπως η παραπάνω.

Υπερφόρτωση τελεστών (σύνταξη)

- Η υπερφόρτωση ενός τελεστή είναι απλά μια συνάρτηση. Αυτό σημαίνει ότι θα πρέπει να έχει τύπο επιστροφής, όνομα και λίστα παραμέτρων.
- Το όνομα της συνάρτησης υπερφόρτωσης ενός τελεστή είναι η σύνθεση της λέξης **operator** και του συμβόλου του τελεστή, για παράδειγμα:

```
operator+, operator++, operator<<, operator==, κ.λπ.
```

- Άρα, η δήλωση της υπερφόρτωσης ενός τελεστή είναι παρόμοια με μια συνάρτηση με τη λέξη operator να αποτελεί τμήμα του ονόματος της συνάρτησης:

```
τύποςΕπιστροφής operatorΣύμβολοΤελεστή(λίστα παραμέτρων);
```

- Παράδειγμα:
  - Η κλάση string έχει υπερφορτωμένο τον τελεστή << έτσι ώστε να μπορεί να παράγει έξοδο λεκτικών χρησιμοποιώντας το cout. Ο μεταγλωττιστής θα καλεί αυτόματα τη συνάρτηση operator<<() όταν χρησιμοποιούμε τον τελεστή << ή θα μπορούμε να καλούμε τη συνάρτηση ρητά.
- Δείτε το example1.cpp

https://github.com/chgogos/oop/blob/master/various/COP3330/lect6/example1.cpp

Λεπτομέρειες σχετικά με την υπερφόρτωση τελεστών (1/2)

- Η υπερφόρτωση τελεστών έχει τους ακόλουθους περιορισμούς:
  - Η υπερφόρτωση ενός τελεστή δεν αλλάζει την προτεραιότητά του.
    - `a + b * c` θα είναι πάντα ισοδύναμο με `a + (b * c)`
  - Η υπερφόρτωση ενός τελεστή δεν αλλάζει την προσεταιριστικότητά του.
    - `a + b + c` θα είναι πάντα ισοδύναμο με `(a + b) + c`
  - Η υπερφόρτωση ενός τελεστή δεν μπορεί να αλλάξει την πληθικότητα του (δηλαδή, τον αριθμό των τελεστέων του)
    - Η συνάρτηση που πραγματοποιεί το a+b θα έχει πάντα 2 παραμέτρους, δεν γίνεται να γίνει η πράξη a+b+c στη συνάρτηση.
  - Δεν είναι δυνατόν να κατασκευαστούν νέοι τελεστές, μόνο νέες εκδόσεις των ήδη υπαρχόντων.
  - Η λειτουργία των τελεστών για τους ενσωματωμένους τύπους δεν αλλάζει.

Λεπτομέρειες σχετικά με την υπερφόρτωση τελεστών (2/2)

- Ορισμένοι τελεστές μπορούν να γραφούν ως συναρτήσεις μέλη μιας κλάσης.
- Ορισμένοι τελεστές μπορούν να γραφούν ως απλές συναρτήσεις, εκτός κλάσης.
  - Συχνά χρησιμοποιείται η δεσμευμένη λέξη friend σε αυτές τις περιπτώσεις.
- Ορισμένοι τελεστές μπορούν να γραφούν και με τους δύο τρόπους και είναι επιλογή του προγραμματιστή.
- Ένας δυαδικός τελεστής έχει δύο ορίσματα (π.χ. f1+f2).
  - Αν γραφεί ως απλή συνάρτηση (που δεν είναι συνάρτηση μέλος της κλάσης), και οι δύο τελεστέοι θα πρέπει να στέλνονται ως παράμετροι, άρα θα πρόκειται για μια συνάρτηση με δύο παραμέτρους.
  - Αν γραφεί ως συνάρτηση μέλος της κλάσης, ο πρώτος τελεστέος θα είναι το καλών αντικείμενο, και ο άλλος τελεστέος θα στέλνεται ως παράμετρος, άρα θα πρόκειται για συνάρτηση με μια παράμετρο.
- Ένας μοναδιαίος τελεστής έχει ένα τελεστέο (π.χ. f++).
  - Αν γραφεί ως απλή συνάρτηση (που δεν είναι συνάρτηση μέλος της κλάσης), ο τελεστέος θα στέλνεται ως παράμετρος.
  - Αν γραφεί ως συνάρτηση μέλος της κλάσης, η συνάρτηση δεν θα δέχεται παράμετρο καθώς θα επενεργεί στο καλών αντικείμενο.

Υπερφόρτωση αριθμητικών τελεστών (1/3)

- Επίδειξη της πρόσθεσης κλασμάτων υπεφορτώνοντας τον τελεστή +.
  - Επιθυμούμε να επιτύχουμε το:

```
Fraction f1, f2(1,2), f3(3,4);
f1 = f2 + f3;
```

  - Η δεύτερη σειρά του παραπάνω κώδικα μεταφράζεται σε:

```
f1 = f2.operator+(f3); // υπερφόρτωση συνάρτησης μέλους

ή

f1 = operator+(f2,f3); // υπερφόρτωση απλής συνάρτησης
```

Υπερφόρτωση αριθμητικών τελεστών (2/3)

- Υπερφόρτωση του + **με συνάρτηση μέλος**. Θα πρέπει να προσθέσουμε την ακόλουθη συνάρτηση στη δήλωση της κλάσης Fraction:

```
Fraction operator+(const Fraction& f) const;
```

- Η δήλωση (υλοποίηση) της συνάρτησης μέλους θα είναι:

```
Fraction Fraction::operator+(const Fraction& f) const {
   Fraction r;
   r.numerator = (numerator * f.denominator)
                  + (f.numerator * denominator);
   r.denominator = (denominator * f.denominator);
   return r;
}
```

https://github.com/chgogos/oop/tree/master/various/COP3330/lect6/fraction_member_overload

Υπερφόρτωση αριθμητικών τελεστών (3/3)

- Υπερφόρτωση του + **με απλή συνάρτηση**. Θα πρέπει να ορίσουμε τη συνάρτηση ως φίλη της κλάσης Fraction έτσι ώστε να επιτραπεί η πρόσβαση στα ιδιωτικά μέλη της κλάσης:

```
friend Fraction operator+(const Fraction &f1, const Fraction &f2);
```

- Η δήλωση (υλοποίηση) της συνάρτησης:

```
Fraction operator+(const Fraction &f1, const Fraction &f2) {
    Fraction r;
    r.numerator = (f1.numerator * f2.denominator) + (f2.numerator * f1.denominator);
    r.denominator = f1.denominator * f2.denominator;
    return r;
}
```

https://github.com/chgogos/oop/tree/master/various/COP3330/lect6/fraction_alone_overload

Υπερφόρτωση συγκριτικών τελεστών

- Οι συγκριτικοί τελεστές, επίσης, μπορούν να υπερφορτωθούν είτε μέσω απλών συναρτήσεων είτε μέσω συναρτήσεων μελών.
  - Η συνάρτηση Equals είχε γραφεί ως εξής:

```
friend bool Equals(const Fraction &x, const Fraction &y);
```

  - Η συνάρτηση αυτή μπορεί να γραφεί υπερφορτώνοντας τον τελεστή ==

```
friend bool operator==(const Fraction &f1, const Fraction &f2);
```

  - Στη συνέχεια παρουσιάζονται οι κλήσεις των συναρτήσεων για κάθε μια από τις παραπάνω περιπτώσεις:

```
Fraction f1, f2;
if (Equals(f1,f2)) cout << “Equals”;
```

vs.

```
Fraction f1, f2;
if (f1==f2)  cout << “Equals”;
```

Υπερφόρτωση συγκριτικών τελεστών

- Για να μπορούν να χρησιμοποιηθούν όλοι οι συγκριτικοί τελεστές, θα χρειαστεί να υπερφορτώσουμε και τους 6:
  - `operator==`
  - `operator!=`
  - `operator<`
  - `operator>`
  - `operator<=`
  - `operator>=`
- Εναλλακτικά μπορούν να υπερφορτωθούν μόνο οι operator== και operator< και να συμπληρωθεί:

```
#include <utility>
using namespace std::rel_ops;
```

Υπερφόρτωση τελεστών stream (1/4)

- Ο τελεστής << (stream insertion=εισαγωγή ροής) και ο τελεστής >> (stream extraction=εξαγωγή ροής) δεν λειτουργούν αυτόματα για αντικείμενα νέων κλάσεων.
- Θα πρέπει να προστεθούν οι συναρτήσεις που πραγματοποιούν υπερφόρτωση των συγκεκριμένων τελεστών εφόσον επιθυμούμε αυτή τη λειτουργικότητα για τις κλάσεις που αναπτύσσουμε.
- Για παράδειγμα:
  - Στην κλάση string ο τελεστής << είναι υπερφορτωμένος ως εξής:

```
ostream& operator<<(ostream &os, const string& s);
```

  - Ωστόσο, τίθενται ορισμένα ερωτήματα:
    - Γιατί η πρώτη παράμετρος είναι ostream&;
    - Γιατί η πρώτη παράμετρος δεν είναι const;
    - Γιατί η συνάρτηση επιστρέφει ένα ostream&, δεν θα έπρεπε να είναι void;
    - Η υπερφορτωμένη συνάρτηση είναι απλή συνάρτηση, θα μπορούσε να είναι συνάρτηση μέλος;

Υπερφόρτωση τελεστών stream (2/4)

- Γιατί η πρώτη παράμετρος είναι ostream&;
  - Το cout είναι ένα αντικείμενο τύπου ostream. Καθώς η κλήση έχει τη μορφή cout << str, η πρώτη παράμετρος στην υπερφόρτωση είναι η cout και η δεύτερη η str.
- Γιατί η πρώτη παράμετρος δεν είναι const;
  - Το cout αναπαριστά την έξοδο στην οθόνη του Η/Υ. Αλλάζοντας την έξοδο στην οθόνη σημαίνει ότι η cout θα πρέπει να μπορεί να αλλάξει, άρα δεν μπορεί να είναι const.
- Γιατί η συνάρτηση επιστρέφει ένα ostream&, δεν θα έπρεπε να είναι void;
  - Αυτό συμβαίνει έτσι ώστε να μπορούν να υποστηριχθούν διαδοχικές κλήσεις (cascading calls). Για παράδειγμα:
    - cout << s1 << s2 << s3; ➔ operator<<(cout, s1) << s2 << s3; ➔ cout << s2 << s3; ➔ operator<<(cout, c2) << s3; ➔ cout << s3; ➔ operator<<(cout, s3) ➔ cout; ➔ ;

Υπερφόρτωση τελεστών stream (3/4)

- Η υπερφορτωμένη συνάρτηση είναι απλή συνάρτηση, θα μπορούσε να είναι συνάρτηση μέλος;
  - Όχι, διότι η πρώτη παράμετρος είναι τύπου ostream, το καλών αντικείμενο θα έπρεπε να είναι ostream (συνεπώς η υπερφορτωμένη συνάρτηση θα έπρεπε να ήταν συνάρτηση μέλος της κλάσης ostream). Καθώς η κλάση ostream δεν έχει αναπτυχθεί από εμάς αλλά αποτελεί τμήμα της τυπικής βιβλιοθήκης της γλώσσας, αυτό δεν μπορεί να γίνει.

Υπερφόρτωση τελεστών stream (4/4)

- Στη συνέχεια παρουσιάζονται τα prototypes για την υπερφόρτωση των τελεστών εισαγωγής και εξαγωγής στην κλάση Fraction.

```
friend ostream& operator<<(ostream& os, const Fraction& f);
friend istream& operator>>(istream& is, Fraction& f);
```

- Και μια πιθανή υλοποίησή τους:

```
ostream &operator<<(ostream &os, const Fraction &f) {
    os << f.numerator << '/' << f.denominator;
    return os;
}
istream &operator>>(istream &is, Fraction &f) {
    char slash;
    is >> f.numerator >> slash >> f.denominator;
    return is;
}
```

- Εκτελέστε το main2.cpp.

https://github.com/chgogos/oop/tree/master/various/COP3330/lect6/fraction_alone_overload

Ερωτήσεις σύνοψης

- Τι είναι η υπερφόρτωση τελεστών;
- Πως μπορεί να γίνει η υπερφόρτωση του τελεστή << σε μια κλάση που αναπτύσσουμε;
- Αναφέρατε τρεις περιορισμούς της υπερφόρτωσης τελεστών.
- Αναφέρατε δύο πιθανούς τρόπους με τους οποίους μπορεί να γίνει η υπερφόρτωση του τελεστή +.

Αναφορές

- http://www.cs.fsu.edu/~xyuan/cop3330/