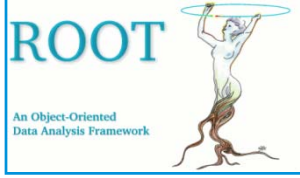
Υπερφόρτωση Τελεστών

- Υπερφόρτωση του τελεστή ανάθεσης τιμής (=)
- Υπερφόρτωση των αριθμητικών τελεστών (+, -, *, /)
- Υπερφόρτωση των αριθμητικών τελεστών ανάθεσης (+=, -=, *=, /=)
- Υπερφόρτωση των συσχετιστικών τελεστών (==)
- Υπερφόρτωση των τελεστών εισόδου-εξόδου (<< και >>)
- Τελεστές μετατροπής casting
- Υπερφόρτωση των τελεστών ++ και --
- Υπερφόρτωση του τελεστή []

Υπερφόρτωση Τελεστών



- Η C++ υποστηρίζει ένα σύνολο τελεστών για τους ενσωματωμένους τύπους της. Δίνει όμως και της δυνατότητα της υπερφόρτωσής τους ώστε να είναι δυνατή η χρήση τους και για τα αντικείμενα νέων κλάσεων.
- Οι παρακάτω τελεστές μπορούν να υπερφορτωθούν με ορισμούς από τον χρήστη:

+	-	*	/	%	^	&
|	~	!	=	<	>	+=
-=	*=	/=	%=	^=	&=	|=
<<	>>	>>=	<<=	==	!=	<=
>=	&&	||	++	--	->*	,
->	[]	()	new	new[]	delete	delete[]

- Μόνο οι τελεστές: ?: :: . και .* δεν είναι δυνατόν να υπερφορτωθούν.
- Ένας δυαδικός τελεστής μπορεί να ορισθεί είτε
  - Με μία συνάρτηση-μέλος η οποία δέχεται ένα όρισμα
  - Με μία συνάρτηση η οποία δεν είναι μέλος της κλάσης και δέχεται δύο ορίσματα. Οι συναρτήσεις αυτές ονομάζονται φίλες (friend). Η φίλη συνάρτηση είναι μια συνάρτηση που δεν είναι μέλος αλλά της παρέχεται πρόσβαση σε όλα τα μέλη της κλάσης στην οποία δηλώνεται
- Ακολουθούν βασικά παραδείγματα υπερφόρτωσης των συνηθέστερων τελεστών.

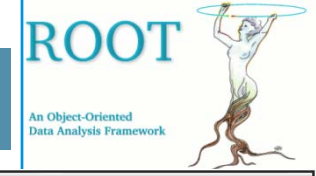
Υπερφόρτωση του τελεστή ανάθεσης τιμής

- Στο παράδειγμά μας βλέπουμε πως μπορούμε να υπερφορτώ-σουμε τον τελεστή ανάθεσης τιμής "=".
- Σημειώνουμε τα ακόλουθα:
  - Παρατηρείστε το όνομα της νέας συνάρτησης operator=
  - Ο δείκτης *this αναφέρεται στο τρέχον αντικείμενο.
  - Επιστρεφόμενος τύπος είναι αναφορά σε ένα αντικείμενο της ίδιας κλάσης. Έτσι λειτουργεί και η αλυσιδωτή ανάθεση πχ. z=y=x;

```
// Paradeigma Yperfortosis telesti =
#include <iostream>
using namespace std;

class Ratio
{ public:
    Ratio(int n=0, int d=1) : num(n), den(d) {  }
    ~Ratio(){ }
    Ratio& operator=(const Ratio&); //Telestis anathesis
    // Ypoloipes dhlwseis
    void print() { cout << num << '/' << den; }
  private:
    int num, den;
    //Ypoloipes dhlwseis
};

Ratio& Ratio::operator=(const Ratio& r)
{ num=r.num;
  den=r.den;
  return *this; //Epistrofi trexontos antikeimenou
}

////////////////////////////////////////////////////////////

int main()
{ Ratio x(22,7), y;

  y=x;

  cout << "y = ";
  y.print();
  cout << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ Ratio_over1_anathesi.cpp
[panos@pc-247 Cpp]$ a.out
y = 22/7
[panos@pc-247 Cpp]$
```

Υπερφόρτωση των αριθμητικών τελεστών



- Η υπερφόρτωση των αριθμητικών τελεστών (+, -, *, /) μπορεί να γίνει όπως στο διπλανό παράδειγμα.
- Στο συγκεκριμένο παράδειγμα γίνεται η υπερφόρτωση του τελεστή πρόσθεσης χρησιμοποιώντας συνάρτηση-μέλος της κλάσης.
- Αυτή η υλοποίηση έχει προβλήματα!!
- Όπως βλέπουμε υλοποιούνται οι προσθέσεις

  z=x+y;

  (Προσοχή αλλάζει και η τιμή του x!!)

  w=y+1;
- Εάν επίσης προσπαθήσουμε την πράξη:

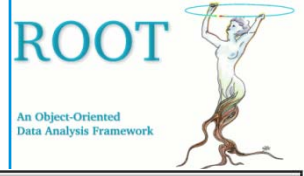
  w=1+x;

  η μεταγλώττιση δεν είναι δυνατή γιατί απλά ο τελεστής + τώρα αφορά πρόσθεση ακεραίων αριθμών!
- Για να ξεπεράσουμε τα προβλήματα επιβάλλεται η χρήση φίλης συνάρτησης όπως στην επόμενη σελίδα.

```
// Paradeigma Yperfortosis aritmitikou telesti +
// xrisimopoiwntas member function
#include <iostream>
using namespace std;

class Ratio
{
  public:
    Ratio(int n=0, int d=1) : num(n), den(d) {  }
    ~Ratio(){ }
    Ratio& operator+(const Ratio&);  // Telestis +
    // Ypoloipes dhlwseis
    void print() { cout << num << '/' << den; }
  private:
    int num, den;
    //Ypoloipes dhlwseis
};

Ratio& Ratio::operator+(const Ratio& a)
{
  num=num*a.den+a.num*den;
  den=den*a.den;
  return *this;
}

////////////////////////////////////////////////////////////

int main()
{ Ratio x(1,2), y(3,5), z, w;

  z=x+y;
  cout << "z = "; z.print(); cout << endl;
  cout << "x = "; x.print(); cout << endl;

  w=y+1;
  cout << "w = "; w.print(); cout << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ Ratio_over1_sum_member.cpp
[panos@pc-247 Cpp]$ a.out
z = 11/10
x = 11/10
w = 8/5
[panos@pc-247 Cpp]$
```

Υπερφόρτωση των αριθμητικών τελεστών



- Η υπερφόρτωση των αριθμητικών τελεστών (+, -, *, /) μπορεί να γίνει εύκολα με χρήση φίλων (friend) συναρτήσεων όπως στο διπλανό παράδειγμα.
- Παρατηρείστε πως οι πράξεις:

  z=x+y;

  w=x+1;

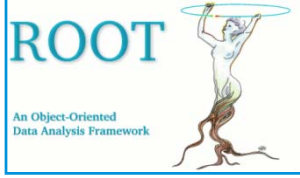
  m=1+x;

  γίνονται όλες κανονικά.

```
// xrisimopoiwntas friend function
#include <iostream>
using namespace std;

class Ratio
{
    Ratio friend operator+(const Ratio&, const Ratio&);
  public:
    Ratio(int n=0, int d=1) : num(n), den(d) { }
    ~Ratio() { }
    // Ypoloipes dhlwseis
    void print() { cout << num << '/' << den; }
  private:
    int num, den;
    //Ypoloipes dhlwseis
};

Ratio  operator+(const Ratio& x, const Ratio& y)
{
  Ratio z(x.num*y.den+y.num*x.den, x.den*y.den);
  return z;
}

////////////////////////////////////////////////////////////

int main()
{ Ratio x(1,2), y(3,5), z, w, m;

  z=x+y;
  w=x+1;
  m=1+x;

  cout << "z = "; z.print(); cout << endl;
  cout << "w = "; w.print(); cout << endl;
  cout << "m = "; m.print(); cout << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ Ratio_over1_sum_friend.cpp
[panos@pc-247 Cpp]$ a.out
z = 11/10
w = 3/2
m = 3/2
[panos@pc-247 Cpp]$
```

Υπερφόρτωση των αριθμητικών τελεστών ανάθεσης

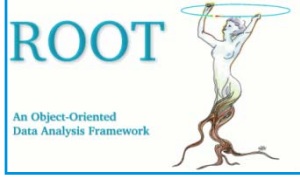


- Η C++ μας επιτρέπει να συνδυάζουμε αριθμητικές πράξεις με τον τελεστή ανάθεσης.
- Οι αριθμητικοί τελεστές ανάθεσης (+=, -=, *=, /=) μπορούν να υπερφορτωθούν όπως στο διπλανό παράδειγμα χρησιμοποιώντας συναρτήσεις-μέλη της κλάσης.

```
// Paradeigma Yperfortosis telesti *=
#include <iostream>
using namespace std;

class Ratio
{ public:
    Ratio(int n=0, int d=1) : num(n), den(d) { }
   ~Ratio(){ }
    Ratio& operator*=(const Ratio& ); //Telestis *=
    // Ypoloipes dhlwseis
    void print() { cout << num << '/' << den; }
  private:
    int num, den;
    //Ypoloipes dhlwseis
};

Ratio& Ratio::operator*=(const Ratio& r)
{ num=num*r.num;
  den=den*r.den;
  return *this;
}

////////////////////////////////////////////////////////////

int main()
{ Ratio x(22,7), y(4,3);

  x*=y;

  cout << "x = ";
  x.print();
  cout << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ Ratio_over1_ar_an.cpp
[panos@pc-247 Cpp]$ a.out
x = 88/21
[panos@pc-247 Cpp]$
```

Υπερφόρτωση των συσχετιστικών τελεστών

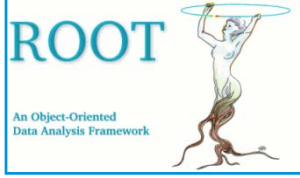


- Οι συσχετιστικοί τελεστές ανάθεσης (<, >, <=, >=, ==, και !=) μπορούν να υπερφορτωθούν με τον ίδιο τρόπο που υπερφορτώνονται οι αριθμητικοί τελεστές, ως φίλες (friend) συναρτήσεις.
- Στο διπλανό παράδειγμα υπερφορτώνεται ο τελεστής ισότητας ==.

```
// Paradeigma Yperfortosis telesti isotitas ==
#include <iostream>
using namespace std;

class Ratio
{
    friend bool operator==(const Ratio&, const Ratio&);
  public:
    Ratio(int n=0, int d=1) : num(n), den(d) { }
    ~Ratio(){ }
    // Ypoloipes dhlwseis
    void print() { cout << num << '/' << den; }
  private:
    int num, den;
    //Ypoloipes dhlwseis
};

bool operator==(const Ratio& x, const Ratio& y)
{ return (x.num*y.den == y.num*x.den); }

////////////////////////////////////////////////////////////

int main()
{ Ratio x(33,21), y(11,7);

  if(x==y) cout << "x equal y" << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ Ratio_over1_sisxet.cpp
[panos@pc-247 Cpp]$ a.out
x equal y
[panos@pc-247 Cpp]$
```

Υπερφόρτωση των τελεστών εξόδου-εισόδου

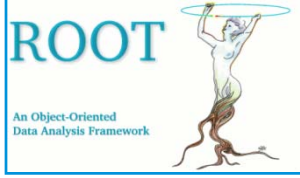


- Η C++ επιτρέπει την υπερφόρτωση των τελεστών εξόδου >> και εισόδου <<.
- Γίνεται χρήση των κλάσεων ostream και istream.
- Στο διπλανό παράδειγμα υπερφορτώνεται ο τελεστής εξόδου >>.
- Παρατηρείστε πόσο απλά μπορεί να γίνει η εκτύπωση ενός αντικειμένου.

```
// Paradeigma Yperfortosis telesti <<
#include <iostream>
using namespace std;

class Ratio
{
    friend ostream& operator<<(ostream&, const Ratio&);
  public:
    Ratio(int n=0, int d=1) : num(n), den(d) { }
    ~Ratio(){ }
    // Ypoloipes dhlwseis
    void print() { cout << num << '/' << den; }
  private:
    int num, den;
    //Ypoloipes dhlwseis
};

ostream& operator<<(ostream& ostr, const Ratio& r)
{ return ostr << r.num << '/' << r.den;}

////////////////////////////////////////////////////////

int main()
{ Ratio x(22,7);
  cout << "x = " << x << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ Ratio_over1_output.cpp
[panos@pc-247 Cpp]$ a.out
x = 22/7
[panos@pc-247 Cpp]$
```

Υπερφόρτωση των τελεστών εισόδου-εξόδου

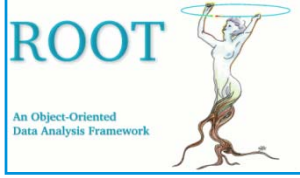


- Στο διπλανό παράδειγμα υπερφορτώνεται τόσο ο τελεστής εξόδου >> όσο και ο τελεστής εισόδου.

```
// Paradeigma Yperfortosis telestwn  << kai >>
#include <iostream>
using namespace std;

class Ratio
{
    friend ostream& operator<<(ostream&, const Ratio&);
    friend istream& operator>>(istream&, Ratio&);
  public:
    Ratio(int n=0, int d=1) : num(n), den(d) { }
    ~Ratio(){ }
    // Ypoloipes dhlwseis
    void print() { cout << num << '/' << den; }
  private:
    int num, den;
    //Ypoloipes dhlwseis
};

ostream& operator<<(ostream& ostr, const Ratio& r)
{ return ostr << r.num << '/' << r.den;}

istream& operator>>(istream& istr, Ratio& r)
{ cout << "\t   Arithmitis: ";  istr >> r.num;
  cout << "\t Paronomastis: ";  istr >> r.den;
  return istr;}

////////////////////////////////////////////////////////////

int main()
{ Ratio x;
  cin >> x;
  cout << "x = " << x << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ Ratio_over1_inout.cpp
[panos@pc-247 Cpp]$ a.out
           Arithmitis: 678
         Paronomastis: 234
x = 678/234
[panos@pc-247 Cpp]$
```

Τελεστές μετατροπής (casting)

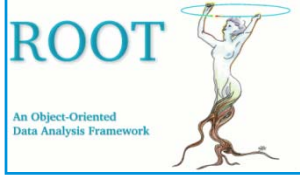


- Για τη μετατροπή από τον τύπο της δεδομένης κλάσης σε άλλο τύπο, είναι απαραίτητη μια διαφορετική συνάρτηση-μέλος η οποία ονομάζεται τελεστής μετατροπής (conversion operator).
- Στο διπλανό παράδειγμα γίνεται η μετατροπή των αντικειμένων Ratio σε μεταβλητές τύπου double.

```
// Paradeigma Yperfortosis casting double
#include <iostream>
using namespace std;

class Ratio
{  public:
     Ratio(int n=0, int d=1) : num(n), den(d) {  }
     ~Ratio(){ }
     operator double() const; //casting double
     // Ypoloipes dhlwseis
     void print() { cout << num << '/' << den; }
  private:
     int num, den;
     //Ypoloipes dhlwseis
};

Ratio::operator double() const
{ return double(num)/den; }

////////////////////////////////////////////////////////////

int main()
{ Ratio x(22,7);

  cout << "x = "; x.print(); cout << endl;

  cout << "x = " << double(x) << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ Ratio_casting.cpp
[panos@pc-247 Cpp]$ a.out
x = 22/7
x = 3.14286
[panos@pc-247 Cpp]$
```

Υπερφόρτωση των τελεστών ++ και --

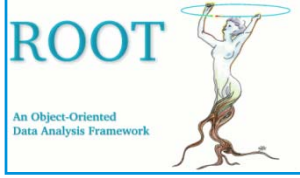


- Ο τελεστής βηματικής αύξησης ++ και μείωσης -- έχουν δύο μορφές ο καθένας: προθεματική και μεταθεματική.
- Στο διπλανό παράδειγμα εικονίζεται ο τρόπος σύνταξης για την υπερφόρτωση του τελεστή ++ (τόσο προθεματική όσο και μεταθεματική).

```
#include <iostream>
using namespace std;

class Ratio
{  public:
     Ratio(int n=0, int d=1) : num(n), den(d) { }
     ~Ratio(){ }
     Ratio operator++();    //Prothematikos
     Ratio operator++(int); //epithematikos
     // Ypoloipes dhlwseis
     void print() { cout << num << '/' << den; }
  private:
     int num, den;
     //Ypoloipes dhlwseis
};

Ratio Ratio::operator++()
{ num=num+den;
  return *this; }

Ratio Ratio::operator++(int)
{ Ratio temp = *this;
  num=num+den;
  return temp; }

////////////////////////////////////////////////////////////

int main()
{ Ratio x(22,7), y, z, t(56,12);

  y=++x;
  z=t++;

  cout << "y = "; y.print(); cout << endl;

  cout << "z = "; z.print(); cout << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ Ratio_over1_pp.cpp
[panos@pc-247 Cpp]$ a.out
y = 29/7
z = 56/12
[panos@pc-247 Cpp]$
```

Υπερφόρτωση του τελεστή []

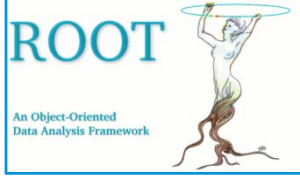


- Το σύμβολο [] υποδηλώνει τον τελεστή αριθμοδείκτη (subscript operator). Το όνομά του προέρχεται από την αρχική χρήση των πινλακων, όπου a[i] αντιπροσωπεύει το μαθηματικό σύμβολο a_i.
- Στο διπλανό παράδειγμα εικονίζεται ο τρόπος σύνταξης για την προσθήκη του τελεστή αριθμοδείκτη στην κλάση.

```
// Paradeigma Yperfortosis array
#include <iostream>
using namespace std;

class Ratio
{   public:
    Ratio(int n=0, int d=1) : num(n), den(d) {  }
    ~Ratio(){ }
    int& operator [] (int); //array!!
    // Ypoloipes dhlwseis
    void print() { cout << num << '/' << den; }
  private:
    int num, den;
    //Ypoloipes dhlwseis
};

int& Ratio::operator [] (int i)
{ if (i==1) return num;
  else return den; }

////////////////////////////////////////////////////////////

int main()
{ Ratio x(22,7);

  cout << "x = "; x.print(); cout << endl;

  cout << "x[1] = " << x[1] << endl;
  cout << "x[2] = " << x[2] << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ Ratio_array.cpp
[panos@pc-247 Cpp]$ a.out
x = 22/7
x[1] = 22
x[2] = 7
[panos@pc-247 Cpp]$
```

Παράδειγμα 1

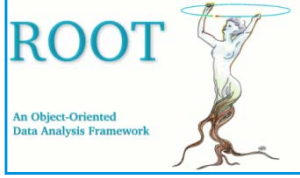


- Αναπτύξτε μια κλάση η οποία αντιπροσωπεύει πρόσωπα.

```
//Paradeigma klasis person
#include <iostream>
using namespace std;
#include <string>

class Person
{
public:
  Person(char* n="", char* nat="U.S.A.", int s=1)
    : name(n), nationality(nat), sex(s) {}
  void printName() { cout << name; }
  void printNationality() { cout << nationality; }
private:
  string name, nationality;
  int sex;
};

int main()
{
  Person creator("Bjarne Stroustrup", "Denmark");
  cout << "The creator of C++ was ";
  creator.printName();
  cout << ", who was born in ";
  creator.printNationality();
  cout << ".\n";
}
```

```
[panos@pc-247 Cpp]$ c++ person.cpp
[panos@pc-247 Cpp]$ a.out
The creator of C++ was Bjarne Stroustrup, who was born in Denmark.
[panos@pc-247 Cpp]$
```

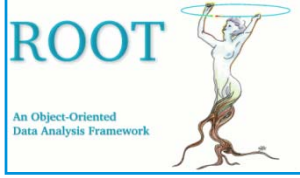
Παράδειγμα 2



- Αναπτύξτε μια κλάση η οποία αντιπροσωπεύει ημερομηνίες.

```
#include <iostream>
using namespace std;
#include <string>

class Date
{
  friend istream& operator>>(istream&, Date&);
  friend ostream& operator<<(ostream&, const Date&);
public:
  Date(int d=0, int m=0, int y=0) : month(m), day(d), year(y) { }
  void setDate(int d, int m, int y) { day = d; month = m; year = y; }
private:
  int day, month, year;
};

istream& operator>>(istream& in, Date& x)
{
  in >> x.day >> x.month >> x.year;
  return in;
}

ostream& operator<<(ostream& out, const Date& x)
{
  static char* monthName[13] = {"","January","February","March","April","May","June","July",
                                "August","September","October","November","December"};
  out << x.day << ' ' << monthName[x.month] << ", " << x.year;
  return out;
}

int main()
{
  Date peace(11,11,1918);
  cout << "World War I ended on " << peace << ".\n";
  peace.setDate(14,8,1945);
  cout << "World War II ended on " << peace <<  ".\n";
  cout << "Enter day, month and year: ";
  Date date;
  cin >> date;
  cout << "The date is " << date << ".\n";
}
```

Παράδειγμα 2




```
[panos@pc-247 Cpp]$
[panos@pc-247 Cpp]$ c++ date.cpp
[panos@pc-247 Cpp]$ a.out
World War I ended on 11 November, 1918.
World War II ended on 14 August, 1945.
Enter day, month and year: 6 9 2009
The date is 6 September, 2009.
[panos@pc-247 Cpp]$
```