



UNIVERSITATEA BABEŞ-BOLYAI
Facultatea de Matematică și Informatică



Programare orientată obiect

Curs 04

Laura Dioşan

POO

□ Clase

- Elemente prietene
- Supraîncărcarea operatorilor

Elemente prietene (friend)

- Se produce o relaxare a încapsulării
 - nu se recomandă folosirea elementelor prietene
 - *o clasă cu prieteni este nesigură*
- Un element prieten cu o clasă poate accesa datele private ale acelei clase
 - funcții prietene
 - funcții globale (funcții externe)

```
friend tip nume_fc(lpf);
```
 - funcții membre ale altei/altor clase

```
friend tip nume_clasă::nume_fc (lpf);
```
 - clase prietene

```
friend nume_clasă;
```
- Relația de prietenie NU este:
 - simetrică
 - tranzitivă

Exemple de elemente prietene

- Funcții prietene
 - Funcții globale (funcții externe)
 - Funcții membre ale altei/altor clase

- Clase prietene

- Exemplu
 - a se consulta directorul *04/friend*
 - *Flower.h, Flower.cpp, test.cpp*

Supraîncărcarea operatorilor

□ Amintiți-vă

- Supraîncărcarea funcțiilor

```
int max(int, int);  
float max(float, float);  
//void max(int, int); //error
```

□ Supraîncărcarea operatorilor

- *syntactic sugar*
- utilizare

pentru un operator @ => o funcție numită **operator@**

- Doar următorii operatori predefiniți pot fi supraîncărcați:
 - +, -, <, ++, +=, ->, ->*, [], (), **new**, **delete**, **sizeof**, ...
- Nu se supraîncarcă:
 - ., *, ::, ?:, #, ##
- Nu se modifică (prin supraîncărcare):
 - precedența operatorilor
 - aritatea operatorilor
 - asociabilitatea operatorilor

Supraîncărcarea operatorilor

- Pt. a supraîncărca un operator @ prin:
 - funcții membre
 - # de param. = aritatea(@) - 1 (din cauza lui *this*)
 - funcții globale (prietene)
 - # de param. = aritatea(@)
- Exemplu
 - a se consulta directorul *04/overload/rational*
 - *Rational.h, Rational.cpp, test.cpp*

Operatorul =

- obiecte simple
 - supraîncărcat implicit
 - se copiază datele membre bit cu bit
- obiecte complexe (cu date dinamice, definite cu pointeri)
 - supraîncărcat explicit
 - același conținut al datelor, dar la diferite adrese de memorie
- a se consulta directorul
04/overload/flower_op=
 - ***Flower.h, Flower.cpp, tests.h, tests.cpp, app.cpp***

Supraîncărcarea operatorilor

| Operatorul @ supraîncărcat ca și | Apel extern (din afara clasei) | Apel intern (din interiorul clasei) |
|----------------------------------|--------------------------------|--|
| Fc. Membră | <i>ob1 @ ob2</i> | <i>ob1.operator@(ob2)</i> |
| Fc. prietenă | <i>ob1 @ ob2</i> | <i>ob1 @ ob2</i> sau <i>operator@(ob1, ob2)</i> |

□ Element membru sau prieten?

- <<, >> - trebuie să fie fc. globale (fc. ne-membre)
- =, (), [], ->, ->* - trebuie să fie fc. membre
- +=, -=, /=, *=, ^=, &=, |=, %=, >, >=, <, <= - fc. membre
- alți operatori – fc. prietene (fc. ne-membre)

Optimizarea valorii returnate

```
Rational Rational::operator+(const Rational& rRight) const{
    Rational rNew;
    rNew.nom = this->nom * rRight.denom + this->denom * rRight.nom;
    rNew.denom = this->denom * rRight.denom;
    rNew.simplify();
    return rNew;
}
```

```
void testPlusReturnValue(){
    Rational r1(1,5);
    Rational r2(3,7);
    cout << " r1 = ";
    printRational(r1);
    cout << "r2 = ";
    printRational(r2);
    cout << "r1+r2 = ";
    printRational(r1 + r2);
}
```

C:\WINDOWS\system32\cmd.exe

```
constructor by parameters Rational : 1 / 5
constructor by parameters Rational : 3 / 7
r1 = 1 / 5
r2 = 3 / 7
r1+r2 = implicit constructor Rational(0,1)
copy constructor Rational from 22/35
destructor Rational for 22/35
22 / 35
destructor Rational for 22/35
destructor Rational for 3/7
destructor Rational for 1/5
```

```
Rational Rational::operator+(const Rational& rRight) const{
    return Rational(this->nom * rRight.denom + this->denom * rRight.nom, this->denom * rRight.denom);
}
```

```
void testPlusReturnValue(){
    Rational r1(1,5);
    Rational r2(3,7);
    cout << " r1 = ";
    printRational(r1);
    cout << "r2 = ";
    printRational(r2);
    cout << "r1+r2 = ";
    printRational(r1 + r2);
}
```

C:\WINDOWS\system32\cmd.exe

```
constructor by parameters Rational : 1 / 5
constructor by parameters Rational : 3 / 7
r1 = 1 / 5
r2 = 3 / 7
r1+r2 = constructor by parameters Rational : 22 / 35
22 / 35
destructor Rational for 22/35
destructor Rational for 3/7
destructor Rational for 1/5
```

Exemplu clase complexe

- a se consulta directorul *04/overload/complexEx*
 - *Flower.h, Gardener.h*
 - *Flower.cpp, Gardener.cpp, test.cpp*

Temă

□ Să se implementeze și să se utilizeze clasa Polinom cu:

- coeficienți întregi
- coeficienți reali

Să se citească mai multe polinoame și să se determine:

- suma lor și
- produsul lor.

Cursul următor

□ Template