Programare orientată obiect

Cursul 4, 5

rev 1.3/PP/20131104

Sumar

- Constructorul de copiere
- Supraîncărcarea operatorilor (1)
  - Concepte de bază
  - Operatorul de atribuire
  - Operatori aritmetici, pe biți, logici și relaționali
- Intrare/Ieșire în C++ (2)
  - Supraîncărcarea operatorilor << și >>

Constructorul de copiere

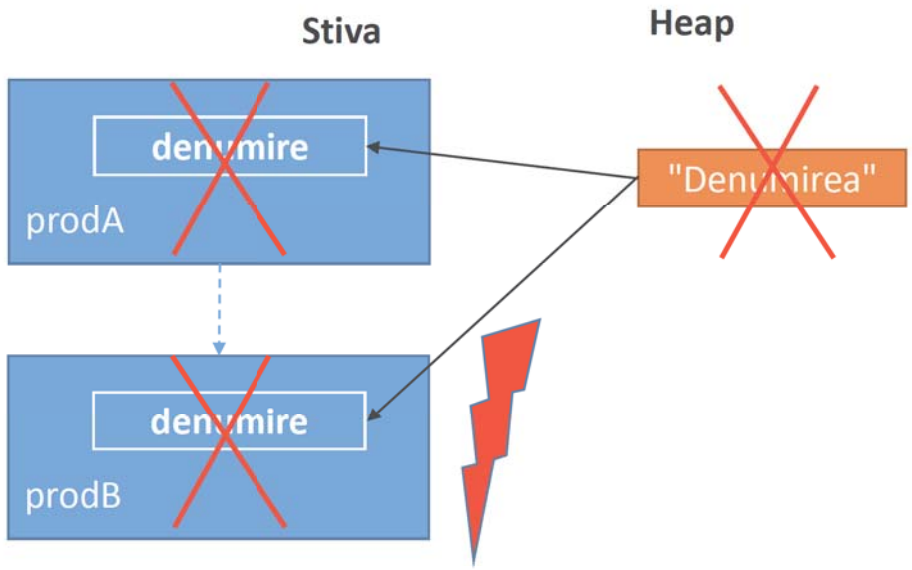
- Definit implicit de către compilator
- Copiere bit cu bit
- Apel implicit:
  1. Inițializarea obiectelor la definire
  2. Obiecte transmise ca parametri prin valoare
  3. Returnarea de obiecte din funcții
- Obligatoriu pentru clasele cu variabile de instanță alocate dinamic!

Clasa Produs

```
class Produs
{
        int cod;
        char *denumire;
        char *um;
        float pret;
//...
};
```

Constructorul de copiere

```
{
  Produs prodA;
  Produs prodB = prodA;
...
}  //destructor prodA
   //destructor prodB
```



Constructorul de copiere

- Prototip:
  - **Clasa(const Clasa &sursa);**
- Parametri
  - Obiectul sursă
- Conținut:
  - Alocare memorie pentru membrii dinamici
  - Copiere conținut membrii sursă -> membrii obiect nou (destinație)
- Returnează
  - Obiectul nou creat

Constructor de copiere

```
Produs::Produs(const Produs &sursa)
{
        cod = sursa.cod;
        denumire = new char[strlen(sursa.denumire) + 1];
        strcpy(denumire, sursa.denumire);
        um = new char[strlen(sursa.um) + 1];
        strcpy(um, sursa.um);
        pret = sursa.pret;
}
```

Supraîncărcarea operatorilor

- Posibilitatea utilizării operatorilor pe tipuri definite de utilizator
- Supraîncărcare pentru majoritatea operatorilor
- Operator – funcție
- Supraîncărcare cu funcții
  - Membre
  - Nemembre (independente, globale), de tip *friend*
    - Nu primesc pointerul this ca parametru -> obligatoriu primesc un obiect al clasei
    - **Obligatoriu**: pentru operatori cu primul operand diferit de tipul clasei
- Comutativitatea nu este asigurată implicit!

Sumar

- Constructorul de copiere
- Supraîncărcarea operatorilor (1)
  - Concepte de bază
  - Operatorul de atribuire
  - Operatori aritmetici, pe biți, logici și relaționali
- Intrare/Ieșire în C++ (2)
  - Supraîncărcarea operatorilor << și >>

Supraîncărcarea operatorilor - Restricții

- Nu pot fi adăugați operatori noi
- Trebuie menținută *aritatea*
- Nu pot fi modificate *prioritatea* și *asociativitatea*
- Operatori care nu pot fi supraîncărcați:
  - Selecție membru (**.**)
  - Selecție membru cu pointer (**.***)
  - Rezoluție (**::**)
  - Condițional (**?:**)
  - Dimensiune (**sizeof**)

Supraîncărcarea operatorilor

- Funcție membră a clasei
  - tip operator **x**();
  - tip operator **x**(param);
- Funcție prietenă, externă clasei
  - tip operator **x**(Clasa ob);
  - tip operator **x**(Clasa ob, tip param);
  - tip operator **x**(**tip param**, Clasa ob);

Clasa Complex

```
class Complex
{
private:
        float re, im;
public:
        Complex();
        …
};
```

Supraîncărcarea operatorilor

- Complex a, b;
- a + b; <=> a.operator+(b);
- ++a; <=> a.operator++();
- 2 + a; <=> operator+(2, a);

Supraîncărcarea operatorilor

- Funcție membră (**obligatoriu**)
  - **()** (funcție), **[]** (indexare), **->** (calificare prin pointer) și **=** (asignare)
- Funcție membră (uzual, **recomandat**)
  - Operație cu asignare (**op=**), operatori unari
- Funcție membră statică (**implicit**) sau individuală (globală)
  - **new** și **delete**
- Funcție membră sau individuală (globală)
  - Alți operatori

Aspecte importante

- Necesitatea
- Tipul funcției:
  - Membră sau globală
- Tipul returnat
- Rezultatul *rvalue* sau *lvalue*

Operatorul de atribuire (=)

- a = b;
- Implicit: copiere bit cu bit
- Situații de avut în vedere:
  - Auto-atribuirea: a = a;
  - Atribuirea înlănțuită: a = b = c;

Operatorul de atribuire (=)

- Prototip:
  - **Clasa & operator=(const Clasa &sursa);**
- Parametri
  - Obiectul sursă
  - Obiectul destinație -> *this*
- Conținut:
  - Eliberare memorie obiect destinație pentru membrii alocați dinamic
  - Alocare memorie pentru membrii alocați dinamic
  - Copiere conținut membrii sursă -> membrii destinație
- Returnează
  - Referință la obiectul destinație (pentru apeluri înlănțuite)

Operatorul de atribire (=)

```
Produs & Produs::operator=(const Produs &sursa)
{
        cod = sursa.cod;
        delete denumire;
        denumire = new char[strlen(sursa.denumire) + 1];
        strcpy(denumire, sursa.denumire);
        delete um;
        um = new char[strlen(sursa.um) + 1];
        strcpy(um, sursa.um);
        pret = sursa.pret;
        return *this;
}
```

Auto-atribuirea??

Clase cu membri alocați dinamic

- Destructor
  - Evitare neeliberare memorie
- Constructor de copiere
  - Evitare eliberare dublă de memorie
- Operatorul de asignare (=)
  - Evitare eliberare dublă de memorie

Operatori aritmetici – unari

- Semn (+, -)
- Pre/post incrementare (++)
  - **tip operator++()**
  - **tip operator++(int)**
- Pre/post decrementare (--)
  - **tip operator--()**
  - **tip operator--(int)**

Operatori aritmetici – unari

Pre-incrementare

```
Complex &operator++();
//…
Complex &Complex::operator++()
{
        re++;
        return *this;
}
```

Post-incrementare

```
Complex operator++(int);
//…
Complex Complex::operator++(int)
{
        Complex temp = *this;

        re++;
        return temp;
}
```

Operatori aritmetici – binari

- Adunare (+)
- Scădere (-)
- Înmulțire (*)
- Împărțire (/)
- Modulo (%)
- Recomandat: supraîncărcarea și a operatorilor de atribuire compusă (+=, -=, *= etc.)

Operatori aritmetici – binari

```
class Complex
{
        //...
        friend Complex  operator+(Complex, Complex);
        friend Complex  operator+(Complex, float);
        friend Complex  operator+(float, Complex);
        Complex &operator+=(Complex);
        //…
};
```

Operatori aritmetici – binari

Complex operator+(float f, Complex z1)

```
{
        Complex tmp;
        tmp.re = f + z1.re;
        tmp.im = z1.im;
        return tmp;
}
```

Complex &Complex::operator+=(Complex z1)

```
{
        re += z1.re;
        im += z1.im;
        return *this;
}
```

Operatori logici

- Unari:
  - Negație (!)
- Binari:
  - ȘI (&&)
  - SAU (||)

Operatori comparație/relaționali

- Egalitate (==)
- Diferit (!=)
- Mai mic (<)
- Mai mic sau egal (<=)
- Mai mare (>)
- Mai mare sau egal(>=)

Operatori comparație/relaționali

```
friend bool operator==(Complex, Complex);
//...
bool operator==(Complex z, Complex z1)
{
        return (z.re == z1.re) && (z.im == z1.im);
}
```

Operatori pe biți

- Unari:
  - Negare (~)
- Binari:
  - ȘI (&)
  - SAU (|)
  - XOR (^)
  - Deplasare la stînga (<<)
  - Deplasare la dreapta (>>)
- Recomadare: supraîncărcarea și a operatorilor de atribuire compusă (&=, |=, <<= etc.)

Mai multe despre intrări/ieșiri în C++

- Dispozitive standard de I/O
  - Clasele **ostream** și **iostream**
- Fișiere
  - Clasele **ofstream** și **ifstream** (`#include <fstream>`)
- Sînt supraîncărcați operatorii **<<**, respectiv, **>>**

Supraîncărcarea operatorilor de I/O

- Se suprîncarcă operatorii << și >>
- Funcții independente, friend
  - Sînt definiți în clasele de I/O din C++
  - Obligatoriu primul parametru este un obiect dintr-o clasa de I/O
- Returnează referințe la obiecte de I/O
  - Pentru înlănțuirea apelurilor în cascadă

Supraîncărcarea operatorilor de I/O

- Operatorul de inserție
  - friend ostream & operator<<(ostream &o, Clasa c);
- Operatorul de extracție
  - friend istream & operator>>(istream &i, Clasa &c);
- Operatorul de inserție (scriere în *fișier*)
  - friend ofstream & operator<<(ofstream &o, Clasa c);
- Operatorul de extracție (citire din *fișier*)
  - friend ifstream & operator>>(ifstream &i, Clasa &c);

Supraîncărcarea operatorilor de I/O

- Exemplu