Laborator 5 – Supraîncărcarea operatorilor

Tema: Clasa `Fractie` $\frac{a}{b}$

Autor: ing. Eduard Donea

Scopul laboratorului

- înțelegerea conceptului de **supraincărcare a operatorilor**;
- folosirea funcțiilor membru și `friend` pentru operatori;
- supraincărcarea operatorilor aritmetici, de comparație și de flux;
- exersarea scrierii de cod curat și testarea în `main`.

Structura fișierelor

- **fractie.h** – declararea clasei `Fractie`;
- **fractie.cpp** – implementarea metodelor și a operatorilor;
- **main.cpp** – testarea funcționalităților definite.

Fișierul `fractie.h`

```
#ifndef FRACTIE_H
#define FRACTIE_H

// Nu uitati sa includeti aici bibliotecile

class Fractie {
    int a; // numarator
    int b; // numitor

public:
    Fractie(int aa = 0, int bb = 1); // constructor cu parametri impliciti

    double getValoare()const;    // returneaza valoarea a/b
    Fractie getInv()const;       // returneaza fractia inversata b/a
    void setdata(int, int);      // seteaza valorile
    float getA()const;           // returneaza numaratorul
    float getB()const;           // returneaza numitorul

    // Operatorii aritmetici
    friend Fractie operator+(const Fractie&, const Fractie&);
    friend Fractie operator-(const Fractie&, const Fractie&);
    friend Fractie operator-(const Fractie&); // unar

    // Operatorul -=
    Fractie& operator-=(const Fractie&);

    // Operatorii de comparatie
    bool operator==(const Fractie&);
    bool operator!=(const Fractie&);
    // Hint: operatorul != poate fi implementat
    // folosind == (ex: return !(*this == x);)

    bool operator>(const Fractie&);
    bool operator>=(const Fractie&);

    // Operatorii de incrementare
    Fractie& operator++();   // prefixat
    Fractie operator++(int); // postfixat

    // Operatorii de flux
    friend istream& operator>>(istream& in, Fractie& z);        // citire
    friend ostream& operator<<(ostream& out, const Fractie& z); // afisare
};

#endif
```

Fișierul `main.cpp` (model)

```cpp
#include "fractie.h"

int main() {

    cout << "=== Testare clasa Fractie ===" << endl;

    // TODO 1: Creati doua obiecte f1 si f2 folosind constructorul
    // cu parametri.
    // Ex: Fractie f1(1, 2), f2(3, 4);

    // TODO 2: Afisati cele doua fractii folosind operatorul <<
    // Ex: cout << f1 << " " << f2;

    // TODO 3: Testati operatorul + (adunare) si afisati rezultatul.
    // Ex: Fractie suma = f1 + f2;

    // TODO 4: Testati operatorul - (scadere).
    // Ex: Fractie produs = f1 - f2;

    // TODO 5: Testati operatorul == si != pentru comparatii.
    // Ex: if(f1 == f2) cout << "Egale"; else cout << "Diferite";

    // TODO 6: Testati operatorii prefixat si postfixat ++.
    // Ex: cout << ++f1 << endl; cout << f1++ << endl;

    // TODO 7: Testati operatorul -= .
    // Ex: f1 -= f2; cout << f1;

    // TODO 8: Cititi o fractie de la tastatura folosind operatorul >>.
    // Ex: Fractie f3; cin >> f3;

    // TODO 9: Afisati valoarea numerica folosind metoda getValoare().
    // Ex: cout << "Valoare: " << f3.getValoare();

    return 0;

}
```

Cerințe (10p)

1. (5p) Implementarea completă a clasei `Fractie` și a operatorilor:
   - operatori aritmetici: `+`, `-`, `-` (unar);
   - operatori de comparație: `==`, `!=`, `>`, `>=`;
   - operatori de incrementare: `++` (prefixat și postfixat);
   - operatori de flux: `>>`, `<<`;
   - metoda `getValoare()` și metoda `getInv()`.
2. (4p) Testarea completă în funcția `main`.
3. (1p) Din oficiu.

Compilare și rulare

```
g++ fractie.cpp main.cpp -o lab5
./lab5
```

Bonus GitHub Classroom

Studenții pot obține până la **3p bonus** dacă încarcă toate laboratoarele până la finalul semestrului, respectiv **1p bonus** dacă încarcă mai mult de jumătate dintre ele.

Pași pentru încărcarea laboratorului:

1. Accesați cursul de pe OCW – Programare Orientată pe Obiecte, secțiunea **Resurse → Bonus GitHub laborator**.
2. Acolo veți găsi linkul **GitHub Classroom** pentru Laboratorul 5:

   `https://classroom.github.com/a/3Zm1gDBB`

3. După acceptare, veți primi propriul repository, ex. `laborator-5-NumePrenume`.
4. Clonați repository-ul local:

```
git clone
↪  git@github.com:Laborator-POO-2025-2026/laborator-5-NumePrenume.git
cd laborator-5-NumePrenume
```

5. După ce ați adăugat fișierele `fractie.h`, `fractie.cpp` și `main.cpp`, urmează comenzile:

```
git add .
git commit -m "Mesaj de commit"
git push origin main
```

Atenție: laboratoarele copiate între studenți sunt considerate nevalide, iar niciunul dintre cei doi participanți nu va primi bonusul. Repository-urile sunt monitorizate automat de sistemul GitHub Classroom.