Laborator 3 – Introducere în Programarea Orientată pe Obiecte

Tema: Studentul Cititor

Autor: ing. Eduard Donea

Poveste

Maria este o studentă pasionată de lectură. În timpul liber citește romane, articole și cărți tehnice, dar deseori uită care au fost ultimele titluri parcurse. Pentru a-și organiza mai bine biblioteca personală, decide să creeze o aplicație care să salveze titlul, autorul, numărul de pagini și nota acordată fiecărei cărți.

Scopul laboratorului

- definirea unei clase în C++;
- constructori și destructor;
- metode `get/set`;
- alocare dinamică și metode de inițializare a obiectelor;
- utilizarea `valgrind` pentru verificarea erorilor de memorie.

Structura fișierelor

- **carte.h** – declararea clasei `Carte`;
- **carte.cpp** – implementarea metodelor clasei;
- **main.cpp** – testarea funcționalităților.

Fișierul `carte.h`

```
#ifndef CARTE_H
#define CARTE_H

/* Aici puteti include si bibliotecile pe care le folositi
pentru un cod mai curat si pentru nu a incarca mult main-ul*/

class Carte {
//private:
    char* titlu;
    char* autor;
    int pagini;
    float rating;

public:
    Carte();
    Carte(const char* titlu, const char* autor, int pagini, float rating);
    ~Carte();

    // metoda de modificare/initializare a obiectului
    void citire();

    void afisare() const;

    void setRating(float rating);
    float getRating() const;

    void setTitlu(const char* titlu);
    const char* getTitlu() const;

};

#endif
```

Fișierul `main.cpp` (model)

```cpp
#include "carte.h"

int main() {

    // TODO 1: Cititi de la tastatura numarul de carti.

    // TODO 2: Alocati dinamic un vector de obiecte Carte.
    // Exemplu: Carte* v = new Carte[n];

    // TODO 3: Pentru fiecare carte, apelati metoda citire()
    // pentru a introduce datele (titlu, autor, pagini, rating).

    // TODO 4: Afisati toate cartile introduse.

    // TODO 5: Sortati cartile dupa rating.
    // Hint: Vom crea un vector auxiliar de indici (0, 1, 2, ... n-1)
    // si vom sorta doar acei indici dupa rating, evitand double-free.

    int* idx = new int[n];
    for (int i = 0; i < n; ++i)
        idx[i] = i;

    // sortare bubble simpla dupa rating
    for (int i = 0; i < n - 1; ++i)
        for (int j = 0; j < n - i - 1; ++j)
            if (v[idx[j]].getRating() > v[idx[j + 1]].getRating()) {
                int temp = idx[j];
                idx[j] = idx[j + 1];
                idx[j + 1] = temp;
            }

    // TODO 6: Afisati vectorul sortat folosind vectorul de indici.

    // TODO 7: Afisati cartea cu ratingul maxim (cea mai apreciata carte).

    // TODO 8: Eliberati memoria alocata dinamic.

    // TODO 9: Rulati programul cu Valgrind pentru a verifica
    // eventualele scurgeri de memorie:
    // g++ -g *.cpp -o lab3
    // valgrind ./lab3

    return 0;
}
```

Cerințe (10p)

1. (4p) Creați o clasă `Carte` care să conțină: `titlu`, `autor`, `pagini`, `rating`. Implementați:
   - constructor fără parametri;
   - constructor cu parametri;
   - destructor;
   - metode `get`/`set`;
   - metoda `afisare()`;
   - metoda `citire()`.
2. (2p) Creați un vector alocat dinamic de tip `Carte`, apelați metoda `citire()` pentru fiecare element și afișați rezultatul.
3. (1.5p) Sortați vectorul crescător după `rating`.
4. (1.5p) Eliberați memoria și verificați cu `valgrind`.
5. (1p) Din oficiu.

Compilare și verificare

```
g++ -g main.cpp carte.cpp -o lab3
./lab3
valgrind ./lab3
```

Mesaje utile Valgrind:

- `definitely lost` → memorie ne-eliberată;
- `invalid read/write` → acces invalid în afara zonelor alocate.

Observație: La rularea cu `valgrind`, nu trebuie să existe mesaje de tip `memory leak detected`.

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 3:

   `https://classroom.github.com/a/opEG5cqC`

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

```
git clone
↪  git@github.com:Laborator-POO-2025-2026/laborator-3-NumePrenume.git
cd laborator-3-NumePrenume
```

5. După ce ați adăugat fișierele `carte.h`, `carte.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.