Laborator 3 – Introducere în Programarea Orientată pe Obiecte

Tema: Studentul Gamer

Autor: ing. Eduard Donea

Poveste

Alex este un student pasionat de jocuri video. Îi place să compare jocurile preferate după gen, studio, numărul de ore jucate și nota pe care o acordă fiecăruia. Pentru a-și organiza colecția, decide să construiască un mic program orientat pe obiecte care să-l ajute să gestioneze informațiile despre jocurile sale video preferate.

Scopul laboratorului

- definirea unei clase în C++;
- constructori și destructori;
- metode `get/set`;
- utilizarea metodelor membre pentru citire și afișare;
- sortarea obiectelor după un criteriu numeric;
- verificarea erorilor de memorie cu `valgrind`.

Structura fișierelor

- **jocvideo.h** – declararea clasei `JocVideo`;
- **jocvideo.cpp** – implementarea metodelor clasei;
- **main.cpp** – testarea funcționalităților.

Fișierul `jocvideo.h`

```
#ifndef JOCVIDEO_H
#define JOCVIDEO_H

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

class JocVideo {
//private
    char* titlu;
    char* studio;
    char* gen;
    int oreJucate;
    float rating;

public:
    JocVideo();
    JocVideo(const char* titlu, const char* studio, const char* gen, int
    ↪  oreJucate, float rating);
    ~JocVideo();

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

    void afisare() const;

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

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

};

#endif
```

Fișierul `main.cpp` (model)

```cpp
#include "jocvideo.h"

int main() {

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

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

    // TODO 3: Pentru fiecare joc, apelati metoda citire()
    // pentru a introduce datele (titlu, studio, gen, oreJucate, rating).

    // TODO 4: Afisati toate jocurile introduse.

    // TODO 5: Sortati jocurile crescator 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 jocul cu ratingul maxim.

    // 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 `JocVideo` care să conțină: `titlu`, `studio`, `gen`, `oreJucate`, `rating`. Implementați:
   - constructor fără parametri;
   - constructor cu parametri;
   - destructor;
   - metode `get`/`set`;
   - metodele `citire()` și `afisare()`.
2. (2p) Creați un vector de obiecte `JocVideo` și afișați toate jocurile introduse.
3. (1.5p) Sortați vectorul crescător după `rating`.
4. (1.5p) Afișați jocul cu `ratingul` cel mai mare.
5. (1p) Eliberați memoria și verificați cu `valgrind`.

Compilare și verificare

```
g++ -g main.cpp jocvideo.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 `jocvideo.h`, `jocvideo.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.