# Proiect 1

Anul Universitar 2023 – 2024, Semestrul II

Programare Orientată pe Obiecte

GRUPA 133

**Cuprins**



Autor: Daniel Wagner

## I. Important înainte să parcurgeți materialul

1) Link https://github.com/mcmarius/poo foarte util in general
2) Link https://github.com/mcmarius/poo/tree/master/tema-1 specific pentru tema/proiectul 1. **Voi presupune că ați citit acest link în continuare, cerințele sunt doar adaptate, și posibil unele lucruri omise.**
3) TODO: De menționat de ChatGPT cum să fie folosit și cum să nu fie (**la proiectul acesta, deloc ideal!!**)

## II. Resurse necesare:

- Conceptele învățate în cadrul cursurilor 1-4
- Laboratoarele 1-3 + posibil ce vom face la Laboratorul 4 (ar fi util să implementați, nu știu dacă ca cerințe obligatorii, ci de ales între și între poate, TBD)
- TODO: de adăugat link-uri oficiale, tutoriale, etc.

**Observație1**: Laboratorul 4 va ilustra, printre altele, compuneri, obligatorii în cadrul proiectului, dar și utilizare de containere STL (**std::vector<T>**, **std::vector<T> std::vector<T>,** etc), recomandate de utilizat în cadrul proiectului, precum și variabile statice.

**Observatie2**: Laboratorul 5 va fi cu moșteniri, care vor intra la proiectul 2 (sau 2+3, dacă alegeți joc/aplicație cu interfață grafică), însă puteți folosi și în cadrul acestui proiect dacă este cazul.

**Observație3**: Cerințele generale obligatorii sunt cele pentru nota 10 maximă (sau mai puțin), în funcție de cât de mult ați implementat din lista de cerințe. Pentru obținerea unei note până la 12, trebuie să îndepliniți criteriile de bonus ilustrate.

## III. Cerințe speciale pentru obținerea notelor maxime (12):

- Codul trebuie să fie bine organizat, cu împărțirea corectă în fișiere **.h** și **.cpp**.
- Utilizarea eficientă a containerelor STL.
- Documentația codului și comentariile relevante pentru metodele și clasele importante.
- TODO: alte cerințe speciale pentru notă maximă? Verificarea corectitudinii datelor de intrare?

# IV. Cerințe generale obligatorii

- Minim 3-4 clase cu sens (pot fi definite toate și în main.cpp, însă nu vă indic), în care să se ilustreze compunere, cc/op=/destructor, getters/setters (doar dacă îi folosiți!), op<<, op>>, dar și câteva metode proprii specifice proiectului ales de voi cu sens în clase (minim 2-3 metode proprii, ideal mai multe), precum și alte concepte învățate la cursuri și laboratoare.
- Toate atributele vor fi definite **private**
- Metodele interne clasei (funcțiile ajutătoare), de asemenea **private** – nu are sens să le poată apela utilizatorul, ci doar voi din interior (TODO: explicații?).
- Utilizare **const** peste tot unde este posibil (în mod special când trimiteți prin referință fără să modificați obiectul sau la metodele care nu modifică obiectul **this**)
- Toate metodele definite trebuiesc și apelate/testate în cadrul claselor sau în **main()**, altfel nu are sens să le fi definit.
- Fără variabile globale!
- **FĂRĂ using namespace std** – niciodata în **.h**, tolerabil în **.cpp**, dar tot neindicat; **Observație**: sunteți liberi să îl folosiți la colocviu, unde orice secundă câștigată conteaza.
- Proiectul să fie încărcat **pe GitHub fără erori de compilare**, cu cât mai multe commit-uri pe parcurs cu mesaje descriptive. Numărul commit-urilor și munca pe parcurs va fi luată în considerare subiectiv la notare în plus. Insă, studenții cu proiecte cu un singur commit sau doar pe final vor fi întrebați mai în amănunt la prezentare.
- README.md care explică proiectul (numele proiectului, scopul proiectului, structura claselor, funcționalități)
- .gitignore pentru fișierele pe care nu doriți să le includeți (nu includeți alte fișiere decât cele necesare proiectului, și **să nu includeți fișiere executabile pe GitHub**, doar cod sursă!!)
- Fișier de input pentru datele citite de la tastatură. Fișierul de input este necesar încât să nu fiu nevoit să introduc manual datele pentru a vă testa proiectul. Puteți să vă redirecționați și voi input-ul din fișier în terminal din același motiv (TODO: explicații cum).

# V. Proiect Exemplu: Sistem de Gestionare a Bibliotecii

## (NU ÎL ALEGEȚI, este ilustrativ, veniți cu propriile propuneri)

1. **Scopul proiectului**:

Crearea unui sistem simplificat pentru gestionarea unei biblioteci, care permite adăugarea, ștergerea, împrumutul și returnarea cărților, precum și gestionarea utilizatorilor bibliotecii.

2. **Descrierea generală**:

Sistemul de gestionare a bibliotecii va folosi conceptele OOP în C++ predate în laboratoare, cum ar fi constructori, destructori, supraîncărcarea operatorilor, utilizarea containerelor STL pentru stocarea datelor (lab4), etc.

3. **Clasele principale**:

i. **Carte**:

- Atribute: titlu, autor, anul publicării, disponibilitate.
- Metode: constructori, getteri/setteri (unde este nevoie doar!), supraîncărcarea operatorilor << și >> pentru afișare și citire.

ii. **Utilizator**:

- Atribute: nume, ID, lista de cărți împrumutate (folosind **std::vector<Carte>**, sau **std::list<Carte>**).
- Metode: constructori, getteri/setteri, metode pentru împrumutul și returnarea unei cărți.

iii. **Biblioteca**:

- Atribute: lista de cărți (folosind **std::vector<Carte>**), lista de utilizatori (folosind **std::vector<Utilizator>**).
- Metode: adăugarea/ștergerea unei cărți, adăugarea/ștergerea unui utilizator, căutarea unei cărți după titlu/autor, împrumutul și returnarea cărților.

4. **Funcționalități suplimentare exemplu (nu neapărat acestea):**

- Supraîncărcarea operatorilor += și -= pentru adăugarea sau ștergerea unei cărți din bibliotecă.
- Utilizarea constructorului de copiere și a operatorului de atribuire pentru gestionarea corectă a copiilor obiectelor. Utilizarea destructorului pentru eliberarea memoriei obiectelor.
- Sistem simplu de autentificare pentru utilizatori: Chiar și în absența unui sistem complex de gestionare a erorilor, se poate implementa un mecanism simplu bazat pe parolă sau pe un cod de acces pentru utilizatori.
- Interfață în linia de comandă: O interfață simplă, care poate include un meniu cu opțiuni pentru utilizator (unde optiunile sunt dintr-un enum) și citirea de la tastatură a acțiunilor dorite, ajută la interacțiunea cu sistemul.

5. **Exerciții și provocări pentru un proiect mai complex**:

- Adăugați funcționalitatea de a salva și încărca starea bibliotecii folosind fișiere.
- Implementați un sistem de notificări pentru cărțile returnate târziu.
- Extindeți sistemul pentru a include recenzii ale cărților adăugate de utilizatori.
- Adaugați și alte clase cu sens în program, dacă doriți.
- Utilizați moștenire unde are sens dacă doriți, însă pe lânga cele 3-4 clase minime cerute

Acest proiect ilustrativ încorporează majoritatea conceptelor discutate în primele patru cursuri și laboratoare și oferă o aplicare practică a lor, pregătindu-vă în același timp pentru proiecte mai complexe și pentru înțelegerea și utilizarea avansată a C++. Însă, proiectul incorporează concepte deja destul de avansate, puteți să vă alegeți ceva mai usor, dar nu banal.

6. **Opțional pentru bonus**:
   Fișier Gr133_*Nume_Prenume*_Proiect1.txt trimis în particular pe Teams cu ce nu ați reușit să implementați și unde v-ați blocat, rezumat general scurt al conceptelor învățate din lucrul în cadrul proiectului, și ce nu s-a înțeles din cerințe, plus ce mai considerați voi relevant. Util pentru feedback dar și autoevaluare.

## VI. **Barem orientativ:** TODO…