

Laborator 8

Tema: Funcții și Clase Template </>

Poveste

O firmă de curierat dorește să implementeze un sistem pentru gestiunea pachetelor. Sistemul trebuie să poată manipula **pachete** cu diverse tipuri de conținut. Fiecare **pachet** are un **ID de colet** și un **conținut** specific. Acest conținut poate fi un obiect complex, cum ar fi un **produs** (cu nume și preț), sau o valoare simplă, precum un cod de voucher de tip **int**.

Acest scenariu este perfect pentru a aplica conceptele de **clase template**. Vom crea o clasă generică **Pachet<T>** care poate stoca un conținut de orice tip **T**. Astfel, putem refolosi aceeași structură pentru a gestiona pachete ce conțin produse electronice, documente sau coduri numerice, demonstrând puterea și flexibilitatea template-urilor în C++.

Scopul laboratorului

- înțelegerea conceptului de **programare generică**;
 - definirea și utilizarea **funcțiilor template**;
 - definirea și instanțierea **claselor template**;
 - organizarea codului template în fișiere header.
-

Structura fișierelor

- **Produs.h / Produs.cpp** — Clasă non-template ce folosește **char*** pentru atribute.
- **Pachet.h** — Clasa template **Pachet<T>** care va fi implementată direct în header.
- **main.cpp** — Testarea funcționalităților.

Fișierul Produs.h

```
#pragma once
#include <iostream>

class Produs {
private:
    char* numeProdus;
    double pret;

public:
    // Metode necesare:
    // - Constructor cu parametri
    // - Destructor
    // - Constructor de copiere
    // - Operator de atribuire
    // - Operator de afisare <<
};
```

Fișierul Pachet.h

```
#pragma once
#include <iostream>

template <class T>
class Pachet {
private:
    char* idColet;
    T continut;

public:
    // Metodele clasei template (implementate direct in .h):
    // - Constructor cu parametri
    // - Destructor
    // - Constructor de copiere
    // - Operator de atribuire
    // - Operator de afisare <<
};
```

Explicații suplimentare

- **Programare Generică** — Permite scrierea de cod (funcții sau clase) care funcționează cu diverse tipuri de date. Acest lucru este realizat în C++ prin **template-uri**. Codul este scris o singură dată, iar compilatorul generează versiunile necesare pentru fiecare tip de date specific utilizat.
- **Clase Template** — O clasă prefixată cu `template <class T>` sau `template <typename T>` devine un șablon. T este un placeholder pentru un tip de date care va fi specificat la crearea unui obiect. De exemplu, `Pachet<Produs>` instruieste compilatorul să genereze o versiune a clasei în care T este înlocuit cu `Produs`.
- **Regula celor Trei** — Deoarece ambele clase, `Produs` și `Pachet`, gestionează memorie alocată dinamic (`char*`), este esențială implementarea corectă a destructorului, constructorului de copiere și operatorului de atribuire pentru a evita probleme precum memory leaks sau double free.

Cerințe (9p + 1p oficiu)

1. Implementare Clasă Non-Template (3p)

- (2p) Implementarea corectă a Regulii celor Trei (destructor, cc, op=) pentru clasa `Produs` pentru a gestiona membrul `char* numeProdus`.
- (1p) Implementarea constructorului și a operatorului de stream pentru `Produs`.

2. Implementare Clasă Template (3p)

- (2p) Implementarea corectă a Regulii celor Trei pentru clasa template `Pachet<T>` (gestionând `char* idColet`), direct în fișierul `.h`.
- (1p) Supraincărcarea corectă a operatorului de stream ca funcție template prietenă, implementată tot în `.h`.

3. Testare în main.cpp (3p)

- (1p) Crearea și afișarea unor obiecte `Pachet<Produs>`.
- (1p) Testarea riguroasă a constructorului de copiere și a operatorului de atribuire pentru ambele clase, pentru a valida managementul memoriei.
- (1p) Instanțierea și utilizarea clasei `Pachet` cu un tip de date de bază (ex. `int`, `float`) pentru a demonstra generalitatea.

4. (1p) Punct din oficiu.

Compilare și verificare

```
g++ -g main.cpp Produs.cpp -o lab8
./lab8
valgrind ./lab8
```

Bonus GitHub Classroom

Studentii 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:

- (a) Accesați cursul de pe OCW – Programare Orientată pe Obiecte, secțiunea **Resurse** → **Bonus GitHub laborator**.
- (b) Acolo veți găsi linkul **GitHub Classroom** pentru Laboratorul 8:

<https://classroom.github.com/a/CeuVjhXY>

- (c) După acceptare, veți primi propriul repository, ex. `laborator-8-NumePrenume`.
- (d) Clonați repository-ul local:

```
git clone git@github.com:Laborator-P00-2025-2026/laborator-8-NumePrenume.git
cd laborator-8-NumePrenume
```

- (e) După ce ați adăugat fișierele corespunzătoare (`Produs.h`, `Produs.cpp`, `Pachet.h`, etc.), urmează comenzile:

```
git add .
git commit -m "Implementare laborator 8 - Functii si Clase Template"
git push origin main
```