****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**Calancea Catalin**

**MI-222**

**Raport**

**pentru lucrarea de laborator Nr.4**

***la cursul de “Programarea Orientată pe Obiect”***

Verificat:

lector universitar

Buldumac Oleg

**Chișinău – 20****23**

**CUPRINS**

[1.INTRODUCERE 3](#_Toc149568596)

[2.SARCINA 4](#_Toc149568597)

[3.CONCLUZIA 5](#_Toc149568598)

[4.WEBOGRAFIE 6](#_Toc149568599)

[5.ANEXĂ 7](#_Toc149568600)

# 1.INTRODUCERE

# Programarea orientată pe obiect (POO) reprezintă un paradigmă esențială în dezvoltarea software-ului, care facilitează organizarea și structurarea codului în mod modular. Moştenirea și compoziția sunt două concepte fundamentale în cadrul POO, oferind programatorilor unelte puternice pentru crearea și gestionarea relațiilor între clase.

# Scopul acestei lucrări de laborator este de a explora în profunzime moştenirea și compoziția în limbajul de programare C++. Vom investiga avantajele și dezavantajele acestor două concepte, analizând modul în care acestea contribuie la dezvoltarea unui cod eficient și ușor de întreținut.

# Un aspect important al acestei lucrări este studierea regulilor de definire a moştenirii și compoziției în C++. Vom explora modalitățile de implementare a acestora, precum și impactul lor asupra structurii și comportamentului programului.

# De asemenea, lucrarea va aborda diferitele forme de moştenire, inclusiv moştenirea publică, protejată și privată, evidențiind avantajele și limitările fiecăreia. Vom explora, de asemenea, conceptul de inițializatori și modul în care aceștia sunt utilizați în cadrul moștenirii și compoziției pentru a asigura o inițializare corespunzătoare a obiectelor.

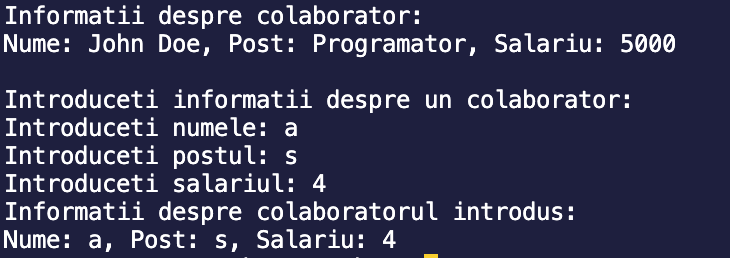
# Principiul de substituție este un alt aspect crucial pe care îl vom investiga, analizând modul în care clasele derivate pot fi utilizate în locul claselor de bază, aducând flexibilitate și extensibilitate în proiecte.

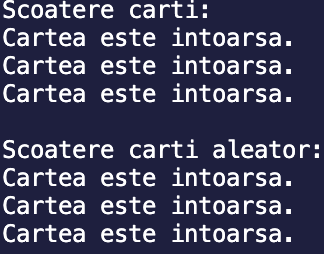
# În final, această lucrare de laborator va oferi o înțelegere detaliată a modului în care moştenirea și compoziția sunt implementate și utilizate în limbajul C++, evidențiind avantajele practice și oferind perspective asupra deciziilor de proiectare.

# 2.SARCINA

а) Să se creeze o ierarhie a claselor om şi colaborator, care ocupă un post anumit şi primeşte un salariu anumit. Să se supraîncarce operatorii pentru ieşiri şi intrări de obiecte, constructorul de copiere, operatorii de atribuire utilizînd funcţiile respective ale clasei de bază.

b) Să se creeze clasa carte de joc, care conţine grad şi culoare. Cartea poate fi întoarsă şi deschisă. Creaţi clasa – butuc de cărţi, care conţine cărţi de joc. Creaţi două clase derivate de la butuc de cărţi, în una cărţile pot fi scoase numai într-o ordine, iar în alta – aleator.

a) 

b) 

# 

# 3.CONCLUZIA

În concluzie, această lucrare de laborator ne-a permis să explorăm în detaliu conceptele de moştenire și compoziție în limbajul de programare C++. Am analizat avantajele și dezavantajele acestor două mecanisme esențiale în programarea orientată pe obiect și am înțeles modul în care acestea contribuie la dezvoltarea unui cod eficient și ușor de întreținut.

Studiul regulilor de definire a moștenirii și compoziției ne-a oferit o perspectivă asupra modului în care putem proiecta relațiile între clase, adaptându-ne la cerințele specifice ale proiectelor noastre. Am învățat să gestionăm forme variate de moștenire și să înțelegem importanța inițializatorilor în asigurarea unei inițializări corespunzătoare a obiectelor în cadrul acestor relații.

Principiul de substituție a devenit clar în ceea ce privește flexibilitatea și extensibilitatea, permițând utilizarea claselor derivate în locul claselor de bază. Acest principiu a demonstrat importanța interoperabilității și schimbului facil între obiecte în cadrul unei ierarhii de moștenire.

În final, această lucrare de laborator ne-a dotat cu cunoștințe solide despre moştenire și compoziție în C++, oferindu-ne instrumente esențiale pentru proiectarea și implementarea eficientă a sistemelor orientate pe obiect. Am înțeles că alegerea între moștenire și compoziție depinde de contextul specific al proiectului, iar o înțelegere clară a avantajelor și limitărilor fiecărei abordări este esențială pentru dezvoltarea software-ului robust și ușor de întreținut.

# 4.BIBLIOGRAFIE

1. Stroustrup, B. (2013). "The C++ Programming Language." Addison-Wesley.
2. Eckel, B. (2003). "Thinking in C++." Prentice Hall.
3. Meyer, B. (1997). "Object-Oriented Software Construction." Prentice Hall.
4. Sutter, H., & Alexandrescu, A. (2005). "C++ Coding Standards: 101 Rules, Guidelines, and Best Practices." Addison-Wesley.

# 5.ANEXĂ

Codul sursa :

Sarcina a ) :

<https://github.com/KetSchnaider/Anul2/blob/main/POO/Lab4/lab4a.cpp>

Sarcina b) :

<https://github.com/KetSchnaider/Anul2/blob/main/POO/Lab4/Lab4b.cpp>