

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**gr. IA-231, Chistol Maxim**

**Raport**

**pentru lucrarea de laborator Nr.5**

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

Verificat:

Kulev Mihail dr., conf. univ.

Departamentul Informatică şi IS,

Facultatea FCIM, UTM

**Chișinău 2024**

**Tema:** Moştenirea multiplă. Clase virtuale

**Scopul lucrării:**

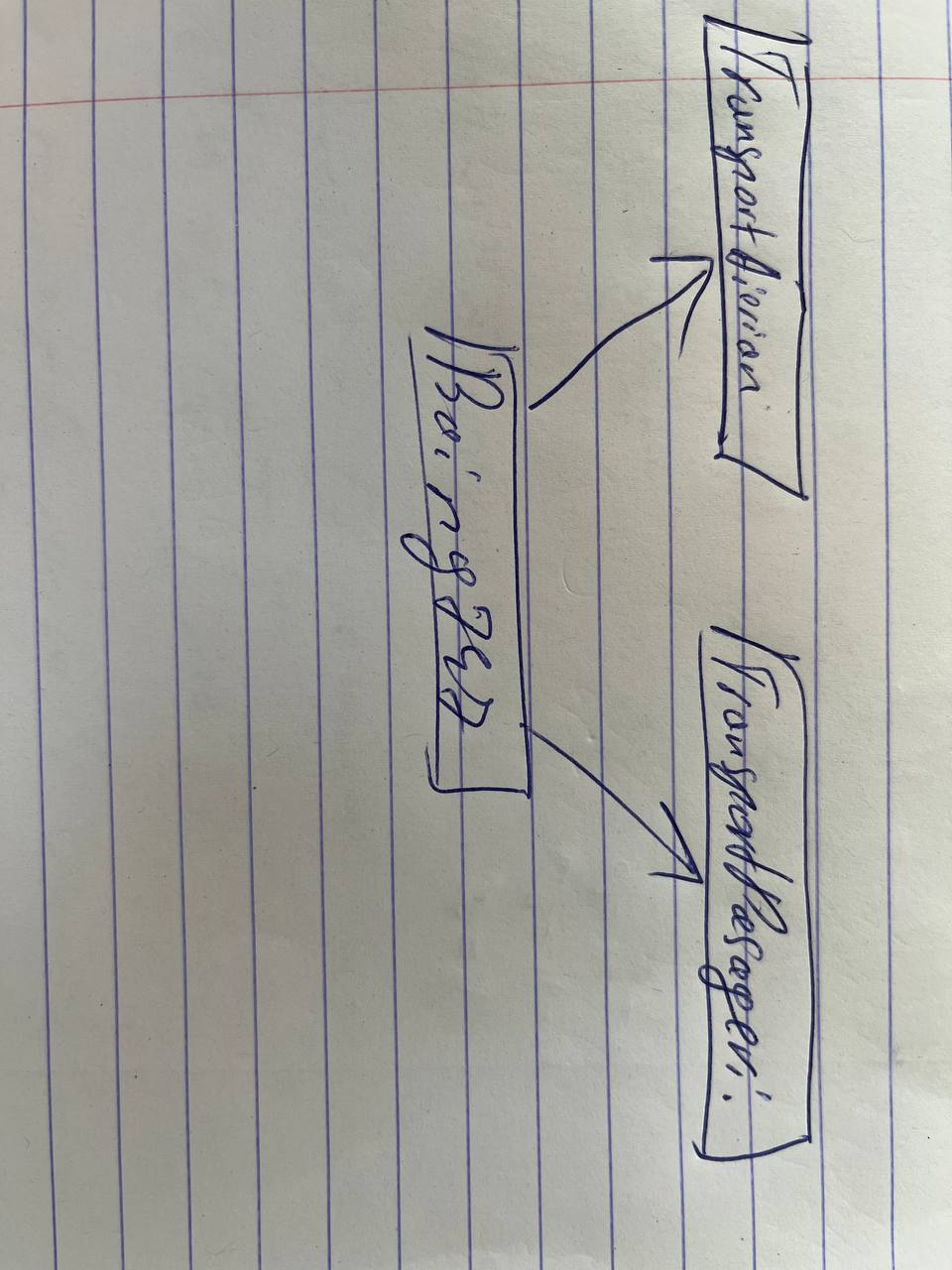
• Studierea regulilor de determinare a moştenirii multiple;

• Studierea avantajelor şi neajunsurilor moştenirii multiple;

• Probleme legate de utilizarea moştenirii multiple;

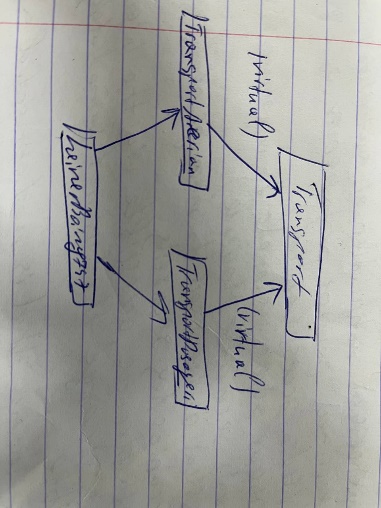
• Studierea rezolvării problemelor;

Motivarea Moştenirea multiplă, reprezintă prin sine moştenirea de la două sau mai multe clase. Pentru a înţelege pentru ce ne trebuie moştenirea multiplă trebuie să ne amintim, că moştenirea simplă nu rezolvă toate problemele, aşa cum câteodată ne obligă să alegem dintre două clase de bază potrivite. Este interesantă părerea lui Buch referitor la acest mecanism: „Moştenirea multiplă este ca o paraşută: de regulă, nu ne trebuie, dar, când întâmplător o să ne trebuiască, o să ne pară rău dacă nu o să fie la îndemână” . În aşa fel, acest mecanism este foarte necesar. Dar, el nu se foloseşte în toate limbajele, dar este realizat în C++. Ca de exemplu: trebuie de descris clasa „Fereastra” cu buton şi denumire, însă clasele Fereastra, Fereastra cu buton, Fereastra cu denumire, sunt deja create şi reprezintă prin sine următoarea ierarhie.



**Figura 1:**Schema pentru [a]

|  |  |
| --- | --- |
| **Codul:** | **Output:** |
| **#include <iostream>**  **#include <string>**  **//Mostenire multipla folosind clase virtuale pentru rezolvarea problemelor ce pot aparea cand se face mostenesc multiple clasele,**  **// Clasa de bază: Transport aerian**  **class TransportAerian {**  **public:**  **TransportAerian(const std::string& tip)//argumentul tip este pentru a seta tipul transportului : tipTransport(tip) {**  **std::cout << "Constructor TransportAerian: " << tipTransport << "\n";**  **}**  **//esteasta functie este virtuala si va fi suprascrisa in clasele derivate**  **virtual void afiseazaInfo() const {**  **std::cout << "Tip transport: " << tipTransport << "\n";**  **}**  **virtual ~TransportAerian() {**  **std::cout << "Destructor TransportAerian\n";**  **}**  **protected:**  **std::string tipTransport;**  **};**  **// Clasa pentru transport de pasageri,unde este o clasa intermidiara ce mosteneste clasa**  **class TransportPasageri : virtual // adauga un atribut suplementar capacitateapasageri public TransportAerian {**  **public:**  **TransportPasageri(const std::string& tip, int capacitate)**  **: TransportAerian(tip), capacitatePasageri(capacitate) {**  **std::cout << "Constructor TransportPasageri\n";**  **}**  **void afiseazaInfo() const override {**  **std::cout << "Tip transport: " << tipTransport << "\n";**  **std::cout << "Capacitate pasageri: " << capacitatePasageri << "\n";**  **}**  **virtual ~TransportPasageri() {**  **std::cout << "Destructor TransportPasageri\n";**  **}**  **protected:**  **int capacitatePasageri;**  **};**  **// Clasa specifică pentru Boeing 747,este o clasa derivata din doua clase transportaeriansipasageri transport,mostenirea multipla Boing mosteneste din ambewle clase,asta ce inseamna ca el poate accesa fucntionalitatile ambelor clase,constructorul claseiinitializeaza pentru a afisa informatiile specifice despre avion**  **class Boeing747 : virtual public TransportAerian, public TransportPasageri {**  **public:**  **Boeing747(int capacitate, double autonomie)**  **: TransportAerian("Transport aerian - Boeing 747"),**  **TransportPasageri("Transport pasageri", capacitate),**  **autonomieZbor(autonomie) {**  **std::cout << "Constructor Boeing747\n";**  **}**  **void afiseazaInfo() const override {**  **TransportPasageri::afiseazaInfo();**  **std::cout << "Autonomie de zbor: " << autonomieZbor << " km\n";**  **}**  **virtual ~Boeing747() {**  **std::cout << "Destructor Boeing747\n";**  **}**  **private:**  **double autonomieZbor;**  **};**  **int main() {**  **Boeing747 avion(416, 14815.0);**  **avion.afiseazaInfo();**  **return 0;**  **}** |  |

****

**Figura 2:**Schema pentru [b]

|  |  |
| --- | --- |
| **Codul:** | **Output:** |
| **#include <iostream>**  **#include <string>**  **// Clasa de bază: Transport**  **class Transport {**  **public:**  **Transport(const std::string& tip) : tipTransport(tip) {**  **std::cout << "Constructor Transport: " << tipTransport << "\n";**  **}**  **void afisare() const {**  **std::cout << "Acesta este un mijloc de transport: " << tipTransport << ".\n";**  **}**  **protected:**  **std::string tipTransport;**  **};**  **// Clasa intermediară: Transport aerian**  **class TransportAerian : public Transport {**  **public:**  **TransportAerian(const std::string& tip, int altitudineMax)**  **: Transport(tip), altitudineMaxima(altitudineMax) {**  **std::cout << "Constructor TransportAerian\n";**  **}**  **void afisare() const {**  **std::cout << "Acesta este un mijloc de transport aerian. Altitudine maximă: "**  **<< altitudineMaxima << " metri.\n";**  **}**  **protected:**  **int altitudineMaxima;**  **};**  **// Clasa intermediară: Transport de pasageri**  **class TransportPasageri : public Transport {**  **public:**  **TransportPasageri(const std::string& tip, int capacitateMax)**  **: Transport(tip), capacitatePasageri(capacitateMax) {**  **std::cout << "Constructor TransportPasageri\n";**  **}**  **void afisare() const {**  **std::cout << "Acesta este un mijloc de transport pentru pasageri. Capacitate: "**  **<< capacitatePasageri << " pasageri.\n";**  **}**  **protected:**  **int capacitatePasageri;**  **};**  **// Clasa derivată: Lainer Boeing 747**  **class LainerBoeing747 : public TransportAerian, public TransportPasageri {**  **private:**  **std::string model;**  **public:**  **LainerBoeing747(const std::string& model, int altitudineMax, int capacitateMax)**  **: TransportAerian("Transport aerian", altitudineMax),**  **TransportPasageri("Transport pasageri", capacitateMax),**  **model(model) {**  **std::cout << "Constructor LainerBoeing747\n";**  **}**  **void afisare() const {**  **TransportAerian::afisare();**  **TransportPasageri::afisare();**  **std::cout << "Acesta este un lainer Boeing 747, model: " << model << ".\n";**  **}**  **};**  **int main() {**  **// Crearea unui obiect de tip LainerBoeing747**  **LainerBoeing747 avion("747-8 Intercontinental", 13000, 416);**  **// Apelarea funcției de afișare prin care si ne va afisa toate datele si ceea ce trebuei sa prezinte**  **avion.afisare();**  **return 0;**  **}** |  |

# **Concluzii:**

# Lucrarea prezentată ilustrează utilizarea moștenirii multiple în C++, o caracteristică esențială pentru construirea ierarhiilor complexe de clase. Am implementat o ierarhie clară pornind de la clasa de bază Transport, continuând cu clasele intermediare TransportAerian și TransportPasageri, și culminând cu clasa derivată specifică LainerBoeing747.

Beneficiile acestei abordări includ reutilizarea eficientă a codului și posibilitatea combinării funcționalităților din mai multe surse. Totuși, moștenirea multiplă poate duce la ambiguități, cum ar fi conflictele între membri moșteniți, rezolvate în acest exemplu prin utilizarea moștenirii virtuale.

Exemplul realizat demonstrează puterea și flexibilitatea moștenirii multiple, oferind o soluție robustă pentru gestionarea relațiilor complexe dintre clase, păstrând totodată un design clar și eficient.

# **Bibliografie:**

1. Lucrare de laborator 5: <https://else.fcim.utm.md/pluginfile.php/125789/mod_resource/content/0/Lab5Rom.pdf> Accesat pe 10.12.2024
2. POO:<https://www.youtube.com/watch?v=rZcTaRU7AAw&list=PLQOaTSbfxUtBm7DxblJZShqBQnBAVzlXX> Acesat pe 10.11.2024
3. Intelegență artificială: <https://chatgpt.com/c/6749b566-c1e8-8001-b9e4-027fa5295f07Accesat> pe 10.11.2024