**MINISTERUL EDUCAŢIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**UNIVERSITATEA TEHNICĂ A MOLDOVEI**

**Facultatea Calculatoare, Informatică și Microelectronică**

**Departamentul Ingineria Software și Automatică**

**RAPORT**

**la lucrarea de laborator nr.4**

**Tema: „Moștenirea multiplă”**

**Disciplina: „Clase și obiecte. Constructori și destructor pentru clasa”**

**A elaborat *st. gr. SI-211, Vozian Vladimir***

**A verificat *lect.univ S. Scrob***

**Chișinău 2022**

**Sarcina**

Varianta 11

*a) Să se creeze, o ierarhie de moştenire: carte, caiet – caiet de notiţe.*

*b) Să se creeze, o ierarhie de moştenire: hârtie - carte, caiet – caiet de notiţe.*

**Codul programului:**

Punctul a)

lab4-a.cpp

#include <iostream>

#include <string>

using namespace std;

class Carte;

class Caiet;

class Caiet\_notite;

class Carte{

private:

    string denumirea;

    string autorul;

public:

    Carte(string denum, string aut){

        denumirea = denum;

        autorul = aut;

    }

    void citire(){

        cout << "Denumirea: " << denumirea << endl;

        cout << "Autorul: " << autorul << endl;

    }

};

class Caiet{

private:

    string continut;

public:

    Caiet(string cont){

        continut = cont;

    }

    void scrie(){

        cout << "Introduceti continutul: " << endl;

        cin >> continut;

    }

    friend class caiet\_de\_notite;

};

class caiet\_de\_notite: public Carte, public Caiet{

public:

    caiet\_de\_notite(string denum, string aut, string cont):Carte(denum, aut), Caiet(cont){

    }

    void citire(){

        Carte::citire();

        cout << "Continutul: " << continut << endl;    }

};

int main(){

caiet\_de\_notite c("Petrea", "Vasile Romanciuk", "continutul initial");

c.citire();

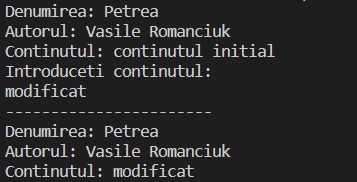
c.scrie();

c.citire();

return 0;

}

Rezultatele:



Punctul b)

lab4-b.cpp

#include <iostream>

#include <string>

using namespace std;

class Hartie;

class Carte;

class Caiet;

class Caiet\_notite;

class Hartie{

private:

    string material;

public:

    Hartie(string mat){

        material = mat;

    }

    void setMaterial(){

        cout << "Dati materialul" << endl;

        cin >> material;

    }

    void afisMaterial(){

        cout << "Materialul: " << material << endl;

    }

};

class Carte: public Hartie{

private:

    string denumirea;

    string autorul;

public:

    Carte(string mat, string denum, string aut):Hartie(mat){

    denumirea = denum;

    autorul = aut;

}

    void citire(){

        cout << "Denumirea: " << denumirea << endl;

        cout << "Autorul: " << autorul << endl;

        afisMaterial();

    }

};

class Caiet: public Hartie{

private:

    string continut;

public:

    Caiet(string mat, string cont):Hartie(mat){

        continut = cont;

    }

    void scrie(){

        cout << "Introduceti continutul: " << endl;

        cin >> continut;

    }

    friend class caiet\_de\_notite;

};

class caiet\_de\_notite: public virtual Carte, public virtual Caiet{

public:

    caiet\_de\_notite(string mat, string denum, string aut, string cont):Carte(mat, denum, aut), Caiet(mat, cont){

    }

    void citire(){

        Carte::citire();

        cout << "Continutul: " << continut << endl;

    }

};

int main(){

caiet\_de\_notite c("bumbac", "Petrea", "Vasile Romanciuk", "continutul initial");

c.citire();

c.scrie();

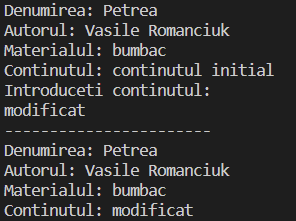
cout << "-----------------------" << endl;

c.citire();

return 0;

}

Rezultatele:



**Concluzii:**

Deci sa lucrat cu moștenirea multiplă in limbajul C++, punctul a) nu era așa greu în realizare fiindca era doar moștenirea a 2 baze de una noua și nu era nevoie de mult lucru la moștenire, însă punctul b) deja a fost o problema fiindcă el reprezintă problema diamantului la moștenire, cand din o clasa bază se creează 2 puțin diferențiate dar asemanatoare după funcțional, și din ele inca o clasa care ar trebui sa moșteneasca metodele și parametrii la amandoua clase dar poate apărea problema daca aceste clase au funcții sau câmpuri cu aceeași denumire și neam gandit la rezolvarea acestei probleme care ar fi prin folosirea operatorului :: pentru a arata ce clasa parinte de folosit, sau de a lucra cu clasele virtuale.