Ministerul Educației și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Colegiul Universității Tehnice a Moldovei

Raport

Lucrare de laborator nr.1

la “Programarea orientată pe obiect”

Tema: „Paradigme de programare. Concepte de bază POO ”

Elaborat de: ***Boico Mihai-Razvan. Daw-241***

Verificat de ***Moraru Magdalena***

Scopul lucrării

Scopul acestei lucrări este familiarizarea cu crearea și utilizarea claselor și obiectelor în C++. De asemenea aplicarea constructori și destructori și lucrul cu modificatori de acces pentru protejarea datelor.

Obiective

1. Aplicarea tipurilor de date de bază pentru a modela obiecte dintr-un domeniu specific;

2. Crearea unei clase care reprezintă un obiect real, cu atribute și metode asociate;

3. Implementarea constructorilor pentru inițializarea obiectelor și a destructorilor pentru eliberarea resurselor;

4. Utilizarea modificatorilor de acces pentru protejarea datelor obiectului;

5. Crearea și utilizarea unor metode pentru operarea asupra obiectelor create (metode pentru a modifica și interoga atributele obiectului).

Sarcina

Cerințele lucrării:

1. Alegerea temei

Fiecare student va alege o temă dintr-un domeniu specific și va crea o clasă care să modeleze obiecte reale. Câteva exemple de teme:

Tema aleasa: Gestionarea unei companii de imobiliare.

2. Crearea claselor și atributelor

Fiecare student va defini o clasă cu minim 5 atribute care descriu obiectul ales. Atributele vor fi protejate sau private iar accesul la ele va fi realizat prin intermediul metodelor publice.

3. Implementarea constructorilor și destructorului

Se vor implementa minim 3 constructori pentru inițializarea atributelor și un destructor pentru afișarea unui mesaj la ștergerea obiectului. Studenții pot alege să implementeze și un constructor implicit, dacă este cazul.

4. Metode pentru operarea asupra obiectelor

Vor fi implementate minim 5 metode publice pentru a lucra cu obiectele create. Aceste metode vor permite interogarea și modificarea atributelor obiectului, în funcție de logica specifică fiecărei entități.

5. *Opțional

Implementarea unui constructor care primește drept argument un alt obiect de la care se va crea un nou obiect, dar cu aceleași atribuite. (Linia 31)

6. *Opțional

Încărcarea codului pe repositoriul Github.

https://github.com/BoicoMihai

codul sursă și rezultatul compilării:

main.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

class Apartment{
    private:
        string location;
        int SqMeter, Floor, Rooms;
        float Price;
        bool Avalability = false;

    public:
        Apartment(){
            location = "unknown";
            SqMeter = 0;
            Floor = 0;
            Rooms = 0;
            Price = 0;
            Avalability = false;
        }

        Apartment(string l, int s, int f, int r, float p, bool A){
            location = l;
            SqMeter = s;
            Floor = f;
            Rooms = r;
            Price = p;
            Avalability = A;
        }

        Apartment(const Apartment& p){
            location = p.location;
            SqMeter = p.SqMeter;
            Floor = p.Floor;
            Rooms = p.Rooms;
            Price = p.Price;
            Avalability = p.Avalability;
        }

        void SetPrice(float p){
            Price = p;
        }

        ~Apartment(){
            cout<<"Obiectul a fost sters!";
            cout<<endl;
        }

        void OutputObject(){
            cout<<"Price: "<<Price<<endl;
            cout<<"Location: "<<location<<endl;
            cout<<"Square meters: "<<SqMeter<<endl;
            cout<<"Rooms: "<<Rooms<<endl;
            cout<<"Avalability: "<< (Avalability ? "Valabil":"Nu este valabil") <<endl;
            cout<<endl;
        }

        void ChangeAvalability(bool p){
            Avalability = p;
        }

```

```
        void OutputPrice(){
            cout<<"Price: "<<Price<<endl;
            cout<<endl;
        }


        void OutputSqmeters(){
            cout<<"Square meters: "<<SqMeter<<endl;
            cout<<endl;
        }
};

int main()
{
    Apartment Ap1;
    Apartment Ap2("Ciocana 12", 58, 3, 2, 182000, true);
    Apartment Ap3(Ap2);

    Ap1.OutputObject();
    Ap2.OutputObject();
    Ap3.OutputObject();

    Ap1.SetPrice(200000);

    Ap1.OutputPrice();
    Ap2.OutputSqmeters();
    Ap3.ChangeAvalability(false);

    Ap3.OutputObject();


    return 0;
}
```

```
Price: 0
Location: unknown
Square meters: 0
Rooms: 0
Avalability: Nu este valabil

Price: 182000
Location: Ciocana 12
Square meters: 58
Rooms: 2
Avalability: Valabil

Price: 182000
Location: Ciocana 12
Square meters: 58
Rooms: 2
Avalability: Valabil

Price: 200000

Square meters: 58

Price: 182000
Location: Ciocana 12
Square meters: 58
Rooms: 2
Avalability: Nu este valabil

Obiectul a fost sters!
Obiectul a fost sters!
Obiectul a fost sters!


...Program finished with exit code 0
Press ENTER to exit console.
```

Concluzia:

Programarea orientată pe obiecte (POO) permite organizarea codului în clase și obiecte care modelează situații reale. În programul dat, clasa Apartament arată concepte de bază POO precum încapsularea, constructorii, destructorul și abstractizarea. Astfel, POO ajută la un cod mai clar, reutilizabil și ușor de întreținut.

Bibliografie:

https://www.geeksforgeeks.org/cpp/object-oriented-programming-in-cpp/

https://www.youtube.com/watch?v=-TkoO8Z07hI