Laborator 1

~ Introducere în C++ și concepte OOP ~

Introducere:

Progamarea Orientată pe Obiecte reprezintă o paradigmă de programare imperativă care are în centrul său conceptul de „obiect". Ea este una dintre cele mai utilizate paradigme din momentul de față. Un cuvânt cheie pe care îl vom folosi de-a lungul lucrului cu obiecte este *class*.

Ce este o clasă? Ce este un obiect?

O clasă este este un șablon care definește structura și comportamentul unor obiecte. Un obiect este o instanță a unei clase. Reprezintă o entitate bazată pe șablonul clasei, dar cu date proprii. Mai concret spus, un obiect este o variabilă de tipul clasei definite.

```
class NumeClasa {
        [modificatori de acces]:
            date;
            metode;
} [nume obiecte de tipul NumeClasa];
```

Class vs Struct:

a. struct (C)
   i. nu pot conține și metode
   ii. modificator de acces **public** *by default*
   iii. nu permite moștenirea
b. struct (C++)
   i. modificator de acces **public** *by default*
   ii. permite moștenirea
c. class (C++)
   i. modificator de acces **private** *by default*
   ii. permite moștenirea

Principiile OOP:

1. **Încapsularea** (Encapsulation)

2. Moștenirea (Inheritance)
3. Abstractizarea (Abstraction)
4. Polimorfismul (Polymorphism)

Vom discuta acum doar despre *încapsulare*, restul rămâne pentru alte laboratoare.

Încapsularea: toate variabilele și funcțiile sunt înglobate într-o singură structură de date (clasă); accesul la anumiți membri ai unei clase poate fi controlat (folosim modificatorii de acces). Încapsularea e asigurată de:

- *Modificatorii de acces*:
  - **private**: datele și metodele NU pot fi accesate din afara clasei
  - **protected**: asemănător cu private, dar mai accesibil (to be continued..)
  - **public**: accesul este permis de oriunde
- *Getters & setters:*
  - **getters**: metode public care întorc valoarea unei date membru private în afara clasei
  - **setters**: metode public care permit modificarea unei date membru private din afara clasei

Constructori:

Constructorul este o metodă specială fără tip returnat (de obicei este public), cu sau fără parametrii și poartă numele clasei, care este apelat în momentul creării unui obiect (adică la declarare). O clasa poate avea mai mulți constructori prin **supraîncărcarea funcțiilor**.

Pe lângă contructor de inițializare, mai există și **constructor de copiere**. Acesta este apelat cu un argument de tipul clasei și copiază conținutul argumentului, fără a-l modifica.

Destructorul se apelează automat în momentul în care obiectul creat va fi distrus. Are ca scop efectuarea unui clean-up (în special a memoriei alocate dinamic)

Definirea unui destructor se face creând o metodă cu același nume ca al clasei precedat de ~.

```
class Student{
  char* name;
  int age;
  public:
    // Definirea constructorului parametrizat
    Student(char* name, int age){
      this->name = name;
      this->age = age;
    }

    // Definirea constructorului de copiere
    Student(const Student& student){
      this->name = student.name;
      this->age = student.age;
    }

    // Definirea unei metode de afisare date
    void displayInfo() {
      std::cout << this->name << std::endl;
      std::cout << this->age << std::endl;
    }

    // Destructor
    ~Student() {
      delete name;
    }
};
```

Orice clasă are implicit dacă nu are deja definit:

- un constructor fără parametrii
- un constructor de copiere ()
- un destructor
- supraîncărcarea operatorului de atribuire (=)

Astfel, următorul cod este valid:

```
class Student{
};

int main() {
    // Instantierea clasei Student, se apelează constructorul fără parametrii
    Student s;

    // Apelarea constructorului de copiere
    Student s2(s);
    Student s3 = s;

    // Apelarea constructorului fără parametrii
    Student s4;

    // Apelarea operatorului de asignare
    s4 = s2;

    // Apelarea destructorului in ordinea inversă a definirii
    // Se apelează destructorul pentru s4
    // Se apelează destructorul pentru s3
    // Se apelează destructorul pentru s2
    // Se apelează destructorul pentru s1
}
```