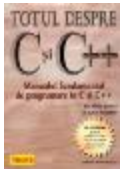
Introducere în Programarea Orientată Obiect (POO)

Mihai Gabroveanu

Bibliografie

- Bjarne Stroustrup: The C++ Programming Language, Adisson-Wesley, 3nd edition, 1997
- Grady Booch, Object-Oriented Analysis and Design with Applications (Second Edition), Adisson-Wesley, 1994
- Jamsa K., Klander L., Totul despre C și C++. Manual fundamental de programare în C și C++, Editura Teora, București, 2000



Bibliografie

- Mirel Coşulschi, Octavian Mustafa, Programare în C++. Concepte moderne şi aplicaţii, Editura Universitaria, 2015
- Liviu Negrescu, Limbajele C și C++ pentru începatori, Vol. II, (editia XI),Editura Albastra, Cluj-Napoca, 2005
- Bruce Eckel, „Thinking in C++, 2nd Edition”, Prentice Hall 2000
  http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html

Detalii Organizatorice

- Curs
  - Prezentarea teoriei generale a Programării Orientate Obiect
- Laborator
  - Implementări practice ale teoriei prezentate la curs în C++
  - Mediul de dezvoltare: Microsoft Visual Studio, Code::Blocks, DevC++
  - *Prezența este obligatorie!*
- Evaluare
  - nota de laborator = NL
    - Doua lucrări
    - Teme
  - examen scris = NE
  - nota finală: (NL+NE)/2

Ce este Programarea Orientată Obiect?

- **Programarea Orientată Obiect (POO)** este o metodă de proiectare și implementare în care programele sunt reprezentate sub forma unor colecții de obiecte (clase) care interacționează între ele prin intermediul mesajelor.
- Limbaje de programare orientate obiect:
  - C++
  - C#
  - Java
  - ...

Concepte de bază în POO

- Principalele concepte care stau la baza POO sunt:
  - Abstractizarea
  - Încapsularea
  - Modularizarea
  - Ierarhizarea

Abstractizarea

- **Abstractizarea** este procesul de grupare a datelor şi metodelor de prelucrare specifice rezolvării unei probleme.
- **Abstractizarea**:
  - exprimă toate caracteristicile esenţiale ale unui obiect care fac ca acesta să se distingă de alte obiecte;
  - oferă o definire precisă a graniţelor conceptuale ale obiectelor din perspectiva unui privitor extern.

Abstractizarea

- Abstractizarea se concentrează asupra caracteristicilor esențiale ale unui obiect, în raport cu perspectiva unui observator.

Source: Grady Booch, Object-Oriented Analysis and Design with Applications (Second Edition)

Abstractizarea - Exemplu

- Tipul abstract de date "**Student**"

```
typedef struct {
    char nume[50];
    char facultatea[30];
    int anStudii;
} Student;
```

- Instanţierea tipului abstract "Student"

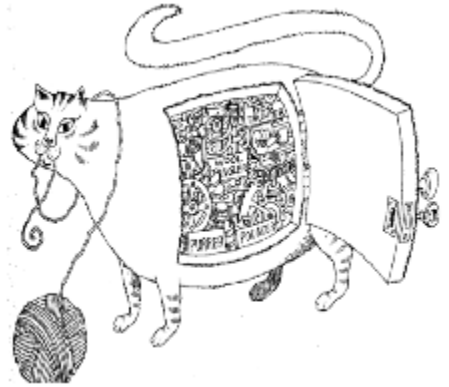
```
Student s={"Popescu Emil","Informatica",1};
```


Încapsularea

- Gruparea datelor şi metodelor aplicabile acestora într-o singură structură de date, definind totodată modul în care obiectul şi restul programului pot referi datele din obiect.
- Concept care defineşte apartenenţa unor proprietăţi şi metode faţă de un obiect.
- Constă în separarea aspectelor externe ale unui obiect care sunt accesibile altor obiecte de aspectele interne ale obiectului care sunt ascunse celorlalte obiecte

Încapsularea

- **Încapsularea** ascunde detaliile implementării unui obiect



Source: Grady Booch, Object-Oriented Analysis and Design with Applications (Second Edition)

Modularizarea

- Modalitatea prin care un program este divizat în subunitati (module) ce pot fi compilate separat
- Un modul grupează abstracţiuni (clase) legate logic între ele

Source: Grady Booch, Object-Oriented Analysis and Design with Applications (Second Edition)

Ierarhizarea

- Reprezintă o ordonare a abstracţiunilor.
- Principalele tipuri sunt:
  - **Moştenirea** (ierarhia de clase) relaţie între clase în care o clasă preia structura şi comportarentul definit în una sau mai multe clase (semantic implică o relaţie de tip "*este un/o*", eng. "*is a*").
  - **Agregarea** (ierarhia de obiecte) relaţie între două obiecte în care unul dintre obiecte aparţine celuilalt obiect. (semantic implică o relaţie de tip "*parte din*", eng. "*part of*")

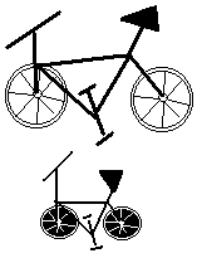
Obiecte

- Un **obiect** este o reprezentare a unei entități din lumea reală asupra căruia se poate întreprinde o acţiune sau care poate întreprine o acţiune
- Un obiect este caracterizat de:
  - *nume*
  - *atribute* (date)
    - valorile atributelor la un moment dat definesc o **stare**
  - *metode* (servicii, operatii)

Clasa

- O **clasă** este o colecție de obiecte cu aceiași structură (caracteristici) și același comportament (metode sau operaţii)

Obiecte - Bicilete



Clasa Biciletă
- atribute
  - tip cadru
  - dimensiunea rotii
  - numar de viteze
- metode
  - accelerează
  - frânează

Identificarea atributelor și metodelor

- În cadrul unei banci un cont bancar are un titular, sold, o rata a dobânzii, numar de cont și se pot efectua operații de depunere, extragere, interogare sold.
- Extragerea atributelor:
  - titular, sold, rata a dobânzii, numar de cont
- Extragerea metodelor:
  - depunere, extragere, interogare sold

Tipuri de date abstracte şi obiecte

- A doua definiţie pentru obiecte şi clase:
  - O **clasa** este o implementare a unui tip de date abstract. Ea defineşte atributele şi metodele care implementează structura de date respectiv operaţiile tipului de date abstract.
  - Un **obiect** este o *instanţă* a unei clase. El este unic determinat de *numele* său şi are o *stare* reprezentată de valorile atributelor sale la un anumit moment particular.

Sintaxa definirii unei clase

- Definirea unei clase constă din două părţi distincte:
  - ***declararea*** clasei
  - ***implementarea*** clasei respective
- Sintaxa

```
class IdNumeClasa {
    declaratii de  date membru;
    declaratii/definiții de functii membru;
};
```


Protecția datelor și funcțiilor membre

- **Funcţiile şi datele unei clase pot fi grupate din punct de vedere al dreptului de acces în:**
  - private **–** date şi funcţii membre care pot fi folosite doar de către funcţiile aparţinând clasei respective;
  - protected **–** similară cu private dar care dă drepturi de acces şi funcţiilor membre ale claselor ce moştenesc clasa respectivă
  - public **–** drept de acces tuturor.

Declararea/definirea unei metode

Declarare

```
class IdNumeClasa{
    tip idNumeMetoda(tip1 p1, ..., tip pn);
};
```

Definire (în afara clasei)

```
tip IdNumeClasa::idNumeMetoda(tip1 p1, ..., tip pn){
    //instructiuni
}
```

Definirea unei metode inline

- **O metodă definită în interiorul clasei se numeste metoda definita inline**

```
class IdNumeClasa{
    tip idNumeMetoda(tip1 p1, ..., tip pn) ){
    //instructiuni
    }
};
```

- **Se recomandă ca o astfel de metodă să conţină un număr redus de intrucţiuni, iar acestea sa nu fie instrucţiuni de ciclare**
- **Functiile declarate ca inline vor fi expandate la compilare, compilatorul generând codul corespunzator functiei în pozitia apelului, în loc să genereze secvenţa de apel.**

Exemplu

Declararea clasei ContBancar

```
class ContBancar{
  private:
      char titular[100];
      char codIBAN[25];
      float sold;
  public:
     void init(char _titular[], char _codIBAN[], float _sold);
     void depune(float suma);
     void retrage(float suma);
     float interogareSold();
     void afisare();

};
```


Exemplu

Implementarea clasei ContBancar

```
void ContBancar::init(char _titular[], char
_codIBAN[], float _sold){
  strcpy(titular,_titular);
  strcpy(codIBAN,_codIBAN);
  sold = _sold;
}

void ContBancar::depune(float suma){
  sold = sold + suma;
}

void ContBancar::retrage(float suma){
  sold = sold - suma;
}
```


Exemplu

Implementarea clasei ContBancar

```
float ContBancar::interogareSold(){
   return sold;
}

void  ContBancar::afisare(){
   printf("Titular: %s\n", titular);
   printf("Cod IBAN: %s\n", codIBAN);
   printf("Sold: %g\n", sold);
}
```

Sintaxa declarării unui obiect

class IdNumeClasa {
  // ....
} *idOb1*, ..., *idObN*;

sau

IdNumeClasa *idOb1*, ..., *idObN*;

Referirea la datele şi metodele membru

idOb.idDataMembru
idPointerOb ->idDataMembru

idOb.idMetodaMembru(lista de parametri);
idPointerOb ->idMetodaMembru(lista de parametri);

Exemplu

```
int main(){
    ContBancar c;
    c.init("Popescu", "RO49RNCB0080005630320001", 100);
    c.afisare();
    printf("Depun 10 RON\n");
    c.depune(10);
    c.afisare();
    printf("Retrag 15 RON\n");
    c.retrage(15);
    c.afisare();
}
```