



Programare orientata pe obiecte

- suport de curs -

**Dobrovat Anca -
Madalina**

**An universitar 2019 – 2020
Semestrul II
Seria 13**

Curs 1

19/02/2020



Generalitati despre curs

1. Curs - miercuri, orele 12-14, Amf. Titeica
2. Laborator - in fiecare saptamana
3. Seminar - o data la 2 saptamani
4. Prezenta la curs/seminar: nu e obligatorie!

Laborator - OBLIGATORIU

- **Examen: 9 iunie 2020 cu seria 14**



Agenda cursului

1. Regulamente UB si FMI
2. Utilitatea cursului de Programare Orientata pe Obiecte
3. Prezentarea disciplinei
4. Primul curs



Agenda cursului

- 1. Regulamente UB si FMI**
2. Utilitatea cursului de Programare Orientata pe Obiecte
3. Prezentarea disciplinei
4. Primul curs



1. Regulamente UB si FMI

Lucruri bine de stiut de studenti:

regulament privind activitatea studenților la UB: <https://www.unibuc.ro/wp-content/uploads/sites/7/2018/07/Regulament-privind-activitatea-profesionala-a-studentilor-2018.pdf>

regulament de etică și profesionalism la FMI:

http://fmi.unibuc.ro/ro/pdf/2015/consiliu/Regulament_etica_FMI.pdf

Se consideră **incident minor** cazul în care un student/ o studentă:
a. preia codul sursă/ rezolvarea unei teme de la un coleg/ o colegă și pretinde că este rezultatul efortului propriu;

Se consideră **incident major** cazul în care un student/ o studentă:
a. copiază la examene de orice tip;

3 incidente minore = un incident major = exmatriculare



1. Regulamente UB si FMI

Lucruri bine de stiut de studenti:

Cazuri

Exemplu:
ELENA, grupa 403, a jucat
e materială este o "copiată" a gândurilor
în locul ocupat din acufitativ. În
continut o parte scursă de o altă
defect, cu o altă acufitativ).
i încaută evenimentul drept incident
major.

Exmatriculare pentru fraudă
- cu drept de înscriere la admitere
- conform consiliului FMI 12.03.2016

B. H. H.



Agenda cursului

1. Regulamente UB si FMI
2. Utilitatea cursului de Programare Orientata pe Obiecte
3. Prezentarea disciplinei
4. Primul curs



2. Utilitatea cursului de Programare Orientata pe Obiecte

Sursa: <https://relus.com/top-10-programming-languages-to-learn-in-2016/>

TOP 10 PROGRAMMING LANGUAGES

1. Java
2. C
3. C++
4. Python
5. C#
6. R
7. PHP
8. JavaScript
9. Ruby
10. Matlab

Orientate Obiect

THE IEEE SPECTRUM SURVEY



2. Utilitatea cursului de Programare Orientata pe Obiecte

Paradigme de programare → Stil fundamental de a programa

Dicteaza:

- **Cum se reprezinta datele problemei** (variabile, functii, obiecte, fapte, constrangeri etc)
- **Cum se prelucreaza reprezentarea** (atribuiri, evaluari, fire de executie, continuari, fluxuri etc)
- **Favorizeaza un set de concepte si tehnici de programare**
- **Influenceaza felul in care sunt ganditi algoritmi de rezolvare a problemelor**
- **Limbaje – in general multiparadigma (ex: Python – imperativ, functional, orientat pe obiecte)**



Agenda cursului

1. Regulamente UB si FMI
2. Utilitatea cursului de Programare Orientata pe Obiecte
3. Prezentarea disciplinei
 - 3.1 Obiectivele disciplinei
 - 3.2 Programa cursului
 - 3.3 Bibliografie
 - 3.4 Regulament de notare si evaluare
4. Primul curs



3. Prezentarea disciplinei

3.1 Obiectivele disciplinei

Curs de programare OO

Ofera o baza de pornire pentru alte cursuri

Obiectivul general al disciplinei:

Formarea unei imagini generale, preliminară, despre programarea orientată pe obiecte (POO).

Obiective specifice:

1. Înțelegerea fundamentelor paradigmei programării orientate pe obiecte;
2. Înțelegerea conceptelor de clasă, interfață, moștenire, polimorfism;
3. Familiarizarea cu șabloanele de proiectare;
4. Dezvoltarea de aplicații de complexitate medie respectând principiile de dezvoltare ale POO;
5. Deprinderea cu noile facilități oferite de limbajul C++.



3. Prezentarea disciplinei

3.2 Programa cursului

1. Principiile programarii orientate pe obiecte
2. Proiectarea ascendenta a claselor. Incapsularea datelor in C++
3. Supraincercarea functiilor si operatorilor in C++
4. Proiectarea descendenta a claselor. Mostenirea in C++
5. Constructori si destructori in C++
6. Modificatori de protectie in C++. Conversia datelor in C++
7. Mostenirea multipla si virtuala in C++
8. Membrii constanti si statici ai unei clase in C++
9. Parametrizarea datelor. Sabloane in C++. Clase generice
10. Parametrizarea metodelor (polimorfism). Functii virtuale in C++. Clase abstracte
11. Controlul tipului in timpul rularii programului in C++
12. Tratarea exceptiilor in C++
13. Recapitulare, concluzii
14. Tratarea subiectelor de examen



3. Prezentarea disciplinei

3.2 Programa cursului

1. Prezentarea disciplinei.

1.1 Principiile programării orientate pe obiecte.

1.2. Caracteristici.

1.3. Programă cursului, obiective, desfășurare, examinare, bibliografie.

2. Recapitulare limbaj C (procedural) și introducerea în programarea orientată pe obiecte.

2.1 Funcții, transferul parametrilor, pointeri.

2.2 Deosebiri între C și C++.

2.3 Supradefinirea funcțiilor, Operații de intrare/ieșire, Tipul referință, Funcții în structuri.



3. Prezentarea disciplinei

3.2 Programa cursului

3. Proiectarea ascendenta a claselor. Incapsularea datelor in C++.

3.1 Conceptele de clasa și obiect. Structura unei clase.

3.2 Constructorii și destructorul unei clase.

3.3 Metode de acces la membrii unei clase, pointerul this. Modificatori de acces în C++.

3.4 Declararea și implementarea metodelor în clasă și în afara clasei.

4. Supraîncărcarea funcțiilor și operatorilor în C++.

4.1 Clase și funcții friend.

4.2 Supraîncărcarea funcțiilor.

4.3 Supraîncărcarea operatorilor cu funcții friend.

4.4 Supraîncărcarea operatorilor cu funcții membru.

4.5 Observații.



3. Prezentarea disciplinei

3.2 Programa cursului

5. Conversia datelor în C++.

5.1 Conversii între diferite tipuri de obiecte (operatorul cast, operatorul= și constructor de copiere).

5.2 Membrii constanți și statici ai unei clase in C++.

5.3 Modificatorul const, obiecte constante, pointeri constanți la obiecte și pointeri la obiecte constante.

6. Tratarea excepțiilor in C++.

7. Proiectarea descendenta a claselor. Mostenirea in C++.

7.1 Controlul accesului la clasa de bază.

7.2 Constructori, destructori și moștenire.

7.3 Redefinirea membrilor unei clase de bază într-o clasa derivată.

7.4. Declarații de acces.



3. Prezentarea disciplinei

3.2 Programa cursului

8. Funcții virtuale în C++.

8.1 Parametrizarea metodelor (polimorfism la executie).

8.2 Funcții virtuale în C++. Clase abstracte.

8.3 Destructori virtuali.

9. Mostenirea multiplă și virtuală în C++

9.1 Moștenirea din clase de bază multiple.

9.2 Exemple, observații.

10. Controlul tipului în timpul rulării programului în C++.

10.1 Mecanisme de tip RTTI (Run Time Type Identification).

10.2 Moștenire multiplă și identificatori de tip (dynamic_cast, typeid).



3. Prezentarea disciplinei

3.2 Programa cursului

11. Parametrizarea datelor. Șabloane în C++. Clase generice

11.1 Funcții și clase Template: Definiții, Exemple, Implementare.

11.2 Clase Template derivate.

11.3 Specializare.

12. Biblioteca Standard Template Library - STL

12.1 Containere, iteratori și algoritmi.

12.2 Clasele string, set, map / multimap, list, vector, etc.



3. Prezentarea disciplinei

3.2 Programa cursului

13. Șabloane de proiectare (Design Pattern)

13.1 Definiție și clasificare.

13.2 Exemple de șabloane de proiectare (Singleton, Abstract Object Factory).

14. Recapitulare, concluzii, tratarea subiectelor de examen.



3. Prezentarea disciplinei

3.3 Bibliografie

1. Herbert Schildt. C++ manual complet. Ed.Teora, Bucuresti, 1997 (si urmatoarele).
 2. Bruce Eckel. Thinking in C++ (2nd edition). Volume 1: Introduction to Standard C++. Prentice Hall, 2000.
 3. Bruce Eckel, Chuck Allison. Thinking in C++ (2nd edition). Volume 2: Practical Programming. Prentice Hall, 2003.
- (Obs: cartile se pot descarca in format electronic, gratuit si legal de la adresa <http://mindview.net/Books/TICPP/ThinkingInCPP2e.html>)
4. Bjarne Stroustrup: The C++ Programming Language, Addison-Wesley, 3rd edition, 1997.
 5. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.



3. Prezentarea disciplinei

3.4 Regulament de notare si evaluare

Curs si laborator: fiecare cu 2 ore pe săptămâna.

Seminar: 1 ora pe saptamana.

Disciplina: semestrul I, avand o durata de desfasurare de 14 săptămâni.

Materia este de nivel elementar mediu si se bazeaza pe cunostintele dobandite la cursul de **Programare procedurala (Programarea calculatoarelor)** din anul I.

Limbajul de programare folosit la curs si la laborator este **C++**.



3. Prezentarea disciplinei

3.4 Regulament de notare si evaluare

Programa disciplinei este impartita in 14 cursuri.

Evaluarea studentilor se face cumulativ prin:

3 lucrari practice (proiecte)

Test practic

Test scris

Toate cele 3 probe de evaluare sunt obligatorii.

Conditiiile de promovare enuntate mai sus se pastreaza la oricare din examenele restante ulterioare aferente acestui curs.



3. Prezentarea disciplinei

3.4 Regulament de notare si evaluare

Regulamentul de laborator este orientativ. Fiecare tutor de laborator are dreptul sa-l adapteze cerintelor grupelor sale!

Cele 3 lucrari practice se realizeaza si se noteaza in cadrul laboratorului, dupa urmatorul program:

Saptamana 1: Test de evaluare a nivelului de intrare.

Saptamana 2: Atribuirea temelor pentru LP1.

Saptamana 3: Consultatii pentru LP1.

Saptamana 4: Predare LP1. **Termen predare LP1: TBA.**

Saptamana 5: Evaluarea LP1.

Saptamana 6: Atribuirea temelor pentru LP2.

Saptamana 7: Consultatii pentru LP2.

Saptamana 8: Predarea LP2. **Termen predare LP2: TBA.**

Saptamana 9: Evaluarea LP2.

Saptamana 10: Atribuirea temelor pentru LP3.

Saptamana 11: Consultatii pentru LP3.

Saptamana 12: Predarea LP3. **Termen predare LP3: TBA.**

Saptamana 13: Evaluarea LP3.

Saptamana 13/14: Test practic de laborator.

Prezenta la laborator in saptamanile 1, 2, 5, 6, 9, 10, 13, 14 pentru atribuirea si evaluarea lucrarilor practice si pentru sustinerea testului practic este obligatorie.



3. Prezentarea disciplinei

3.4 Regulament de notare si evaluare

Regulamentul de laborator este orientativ. Fiecare tutor de laborator are dreptul sa-l adapteze cerintelor grupelor sale!

Consultatiile de laborator se desfasoara pe baza intrebarilor studentilor.

Prezenta la laborator in saptamanile 3, 4, 7, 8, 11, 12 pentru consultatii este recomandata, dar facultativa.

Lucrarile practice se realizeaza individual.

Notarea fiecărei lucrări practice se va face cu note de la 1 la 10.

Atribuirea temelor pentru lucrarile practice se face prin prezentarea la laborator in saptamana precizata mai sus sau in oricare din urmatoarele 2 saptamani. **Indiferent de data la care un student se prezinta pentru a primi tema pentru una dintre lucrarile practice, termenul de predare a acesteia ramane cel precizat in regulament.** In consecinta, tema pentru o lucrare practica nu mai poate fi preluata dupa expirarea termenului ei de predare.



3. Prezentarea disciplinei

3.4 Regulament de notare si evaluare

Regulamentul de laborator este orientativ. Fiecare tutore de laborator are dreptul sa-l adapteze cerintelor grupelor sale!

Predarea lucrarilor practice se face la adresa indicata de tutorele de laborator, inainte de termenele limita de predare, indicate mai sus pentru fiecare LP.

Dupa expirarea termenelor respective, lucrarea practica se mai poate trimite prin email pentru o perioada de gratie de 2 zile (48 de ore).

Pentru fiecare zi partiala de intarziere se vor scadea 2 puncte din nota atribuita pe lucrare.

Dupa expirarea termenului de gratie, lucrarea nu va mai fi acceptata si va fi notata cu 1.



3. Prezentarea disciplinei

3.4 Regulament de notare si evaluare

Nota laborator = medie aritmetica a celor 3 note obtinute pe proiecte.

Pentru evidentierea unor lucrari practice, tutorele de laborator poate acorda un bonus de pana la 2 puncte la nota pe proiecte astfel calculata.

Studentii care nu obtin cel putin nota 5 pentru activitatea pe proiecte nu pot intra in examen si vor trebui sa refaca aceasta activitate, inainte de prezentarea la restanta.



3. Prezentarea disciplinei

3.4 Regulament de notare si evaluare

Testul practic (Colocviu) - in saptamana 14

- Consta dintr-un program care trebuie realizat individual intr-un timp limitat (90 de minute) si va avea un nivel mediu.
- Notare: de la 1 la 10, conform unui barem anuntat odata cu cerintele.

Testul practic este obligatoriu.

Studentii care nu obtin cel putin nota 5 la testul practic de laborator nu pot intra in examen si vor trebui sa il dea din nou, inainte de prezentarea la restanta.



3. Prezentarea disciplinei

3.4 Regulament de notare si evaluare

Testul scris:

Consta dintr-un set de 18 intrebari

- 6 intrebari de teorie
- 12 intrebari practice.

Notarea testului scris se va face cu o nota de la 1 la 10 (1 punct din oficiu si cate 0,5 puncte pentru fiecare raspuns corect la cele 18 intrebari).

Studentii nu pot lua examenul decat daca obtin cel putin nota 5 la testul scris.



3. Prezentarea disciplinei

3.4 Regulament de notare si evaluare

Examenul se considera luat daca studentul respectiv a obtinut **cel putin nota 5 la fiecare dintre cele 3 evaluari (activitatea practica din timpul semestrului, testul practic de laborator si testul scris).**

In aceasta situatie, nota finala a fiecarui student se calculeaza ca medie ponderata intre notele obtinute la cele 3 evaluari, ponderile cu care cele 3 note intra in medie fiind:

25% - nota pe lucrarile practice (proiecte)

25% - nota la testul practic

50% - nota la testul scris

Seminar - **maxim 1p care se adauga la nota de la testul scris, daca si numai daca, nota de la testul scris ≥ 5 .**



Statistici 2018 - 2019

	Număr studenți	Procent din total studenți	Procent din total prezenți
Total	365		
Prezenți	322	87.95%	
Absenți	44	12.05%	
Trecuți	169	46.30%	52.65%
Picați	196	53.70%	61.06%
Nota 10	14	3.84%	4.36%
Nota 9	24	6.58%	7.48%
Nota 8	47	12.88%	14.64%
Nota 7	60	16.44%	18.69%
Nota 6	21	5.75%	6.54%
Nota 5	3	0.82%	0.93%
Nota 4	152	41.64%	47.35%
Nota 3	0	0.00%	0.00%



Statistici 2015 - 2016

	<u>Numar studenti</u>	Procent din total studenti	Procent din total prezenti
Total	268		
Prezenti	219	81.71%	
Absenti	49	18.29%	
Trecuti	95	35.44%	43.37%
Picati	173	64.55%	56.62%
Nota 10	19	7.00%	8.67%
Nota 9	23	8.50%	10.50%
Nota 8	37	13.80%	16.90%
Nota 7	36	13.43%	16.43%
Nota 6	10	3.73%	4.56%
Nota 5	0	0%	0%
Nota 4	124	46.27%	56.62%
Nota 3	0	0%	0%



Statistici 2009 - 2010

	<u>Numar studenti</u>	Procent din total studenti	Procent din total prezenti
Total	<u>Aprox. 200</u>		
Prezenti	157	<u>Aprox. 70%</u>	
Absenti	<u>Aprox. 40</u>	<u>Aprox. 30%</u>	
Trecuti	83	40%	53%
Picati	74	37%	47%
Nota 10	2	1%	1.30%
Nota 9	20	10%	12.75%
Nota 8	26	13%	16.56%
Nota 7	24	12%	15.29%
Nota 6	6	3%	3.82%
Nota 5	0	0%	0%
Nota 4	25	12.5%	15.92%
Nota 3	54	27%	34.40%



Agenda cursului

1. Regulamente UB si FMI
2. Utilitatea cursului de Programare Orientata pe Obiecte
3. Prezentarea disciplinei
4. Primul curs
 - 4.1 Completări aduse de limbajul C++ față de limbajul C
 - 4.2 Principiile programarii orientate pe obiecte



4. Curs 1

4.1 Completări aduse de limbajul C++ față de limbajul C

Intrări și ieșiri

Limbajul C++ furnizează obiectele cin și cout, în plus față de funcțiile scanf și printf din limbajul C. Pe lângă alte avantaje, obiectele cin și cout nu necesită specificarea formatelor.

// operator este o functie care are ca nume un simbol (sau mai multe simboluri)

```
int main()
```

```
{int x,y,z;
```

```
    cin >>x; // operatorul >>(cin,x) care intoarce fluxul (prin referinta ) cin din care s-a  
extras data x
```

```
    cin>>y>>z; //este de fapt >>(>>(cin,y), z)
```

```
cout<<x; // operatorul <<(cout, x ) -intoarce fluxul cout (prin referinta) in care s-a inserat x
```

```
cout<<y<<z; // este de fapt <<(<<(cout,y),z) -afiseaza y si z
```



4. Curs 1

4.1 Completări aduse de limbajul C++ față de limbajul C

Supraîncărcarea funcțiilor (un caz de *Polimorfism la compilare*)

Limbajul C++ permite utilizarea mai multor funcții care au același nume, caracteristică numită supraîncărcarea funcțiilor. Identificarea lor se face prin numărul de parametri și tipul lor.

Exemplu:

```
void afis (int a)
{
    cout<<"int"<<a;
}
```

```
void afis (char a)
{
    cout<<"char"<<a;
}
```



4. Curs 1

4.1 Completări aduse de limbajul C++ față de limbajul C

Funcții cu valori implicite

Într-o funcție se pot declara valori implicite pentru unul sau mai mulți parametri. Atunci când este apelată funcția, se poate omite specificarea valorii pentru acei parametri formali care au declarate valori implicite.

Valorile implicite se specifică o singură dată în definiție (de obicei în prototip).

Argumentele cu valori implicite trebuie să fie amplasate la sfârșitul listei.

Exemplu:

```
void Adunare (int a=5, double b = 10) { ... ; }
```

...

```
Adunare ();    // <=> Adunare (5, 10);
```

```
Adunare (1);   // <=> Adunare (1, 10);
```

```
Adunare (1, 4); // <=> Adunare (1, 4);
```



4. Curs 1

4.1 Completări aduse de limbajul C++ față de limbajul C

Alocare dinamica

```
int *pi;  
pi=new int;
```

delete pi; // elibereaza zona adresata de pi -o considera neocupata

pi=**new** int(2); // alocă zona și initializează zona cu valoarea 2

pi=**new** int[2]; // alocă un vector de 2 elemente de tip întreg

delete [] pi; //eliberează întreg vectorul

//-pentru new se folosește delete

//- pentru new [] se folosește delete []



4. Curs 1

4.1 Completări aduse de limbajul C++ față de limbajul C

Tipul referinta

```
int i;  
int *pi,j;
```

int & ri=i; //ri este alt nume pentru variabila i

pi=&i; // pi este adresa variabilei i

*pi=3; //in zona adresata de pi se pune valoarea 3

O referință este un alt nume al unui obiect (variabila).

Pentru a putea fi folosită, o referință trebuie inițializată în momentul declarării, devenind un alias (un alt nume) al obiectului cu care a fost inițializată.



4. Curs 1

4.1 Completări aduse de limbajul C++ față de limbajul C

Tipul referinta

Restricții pentru referințe:

1. o referință trebuie să fie inițializată când este definită, dacă nu este membră a unei clase, un parametru de funcție sau o valoare returnată;
1. nu se poate referi o altă referință;
1. nu se poate crea un pointer către o referință;
1. referințele nule sunt interzise.



4. Curs 1

4.1 Completări aduse de limbajul C++ față de limbajul C

Transmiterea parametrilor

```
void f(int a, int *b, int &c)
```

```
{
```

```
    a++; // modificarile facute asupra variabilei a nu modifica parametrul  
    actual x;
```

```
    (*b)++; // se modifica variabila adresata de pi deci parametrul actual y
```

```
    c++; // modifica parametrul actual z; pentru c nu se rezerva zona noua de  
    memorie
```

```
}
```

```
int main()
```

```
{    int x,y,z;
```

```
    x=y=z=0;
```

```
    f(x,&y,z) ;
```

```
}
```



4. Curs 1

4.2 Principiile programarii orientate pe obiecte

Principalele concepte (caracteristici) ale POO sunt:

Obiecte

Clase

Mostenire

Ascunderea informatiei

Polimorfism

Sabloane



4. Curs 1

4.2 Principiile programarii orientate pe obiecte

O **clasa** definește atribute și metode.

```
class X{  
    //date membre  
    //functii membre  
};
```

- mentioneaza proprietatile generale ale obiectelor din clasa respectiva
- clasele nu se pot “rula”
- folositoare la encapsulare (ascunderea informatiei)
- reutilizare de cod: mostenire



4. Curs 1

4.2 Principiile programarii orientate pe obiecte

Un **obiect** este o instanta a unei clase care are o anumita stare (reprezentata prin valoare) si are un comportament (reprezentat prin functii) la un anumit moment de timp.

- au stare si actiuni (metode/functii)
- au interfata (actiuni) si o parte ascunsa (starea)
- Sunt grupate in clase, obiecte cu aceleasi proprietati

Un **program orientat obiect** este o colectie de obiecte care interactioneaza unul cu celalalt prin mesaje (aplicand o metoda).



4. Curs 1

4.2 Principiile programarii orientate pe obiecte

Principalele concepte (caracteristici) ale POO sunt:

Incapsularea – contopirea datelor cu codul (metode de prelucrare si acces la date) în clase, ducând la o localizare mai bună a erorilor și la modularizarea problemei de rezolvat;

Moștenirea - posibilitatea de a extinde o clasa prin adaugarea de noi functionalitati;

- multe obiecte au proprietati similare
- reutilizare de cod



4. Curs 1

4.2 Principiile programarii orientate pe obiecte

Principalele concepte (caracteristici) ale POO sunt:

Ascunderea informatiei

foarte importanta

public, protected, private

Avem acces?	public	protected	private
Aceeasi clasa	da	da	da
Clase derivate	da	da	nu
Alte clase	da	nu	nu



4. Curs 1

4.2 Principiile programarii orientate pe obiecte

Principalele concepte (caracteristici) ale POO sunt:

Polimorfismul (la executie – *discutii mai ample mai tarziu*) – într-o ierarhie de clase obtinuta prin mostenire, o metodă poate avea implementari diferite la nivele diferite in acea ierarhie;

Sabloane

- cod mai sigur/reutilizare de cod
- putem implementa lista inlantuita de
 - intregi
 - caractere
 - float
 - obiecte



Perspective

1. Se vor discuta directiile principale ale cursului, feedback-ul studentilor fiind hotarator in acest aspect

- intelegerea notiunilor
- intrebari si sugestii

2. Cursul 2:

- Introducere in OOP. Clase. Obiecte