

FIȘA DISCIPLINEI

Bazele programării orientate obiect
Anul universitar 2025-2026

1. Date despre program

1.1. Instituția de învățământ superior	Universitatea Babeș-Bolyai Cluj-Napoca
1.2. Facultatea	Facultatea de Matematică și Informatică
1.3. Departamentul	Departamentul de Informatică
1.4. Domeniul de studii	Matematică / Informatică
1.5. Ciclu de studii	Licență
1.6. Programul de studii / Calificarea	Matematică - Informatică
1.7. Forma de învățământ	Cu frecvență

2. Date despre disciplină

2.1. Denumirea disciplinei		Bazele programării orientate obiect				Codul disciplinei		MLR5234			
2.2. Titularul activităților de curs			Prof. Dr. Dioșan Laura								
2.3. Titularul activităților de seminar			Prof. Dr. Dioșan Laura								
2.4. Anul de studiu		1	2.5. Semestrul		2	2.6. Tipul de evaluare		E	2.7. Regimul disciplinei		Obligatoriu

3. Timpul total estimat (ore pe semestru al activităților didactice)

3.1. Număr de ore pe săptămână	5	din care: 3.2. curs	2	3.3. seminar/ laborator/proiect	3
3.4. Total ore din planul de învățământ	70	din care: 3.5. curs	28	3.6 seminar/laborator/proiect	42
Distribuția fondului de timp pentru studiul individual (SI) și activități de autoinstruire (AI)					ore
Studiul după manual, suport de curs, bibliografie și notițe (AI)					30
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren					25
Pregătire seminare/ laboratoare/ proiecte, teme, referate, portofolii și eseuri (mai mare sau egal cu nr. total ore prevăzut în calendarul disciplinei pentru temele de control)					20
Tutoriat (consiliere profesională)					2
Examinări					3
Alte activități [de ex.: comunicare bidirecțională cu titularul de disciplină / tutorele]					
3.7. Total ore studiu individual (SI) și activități de autoinstruire (AI)				80	
3.8. Total ore pe semestru				150	
3.9. Numărul de credite				6	

4. Precondiții (acolo unde este cazul)

4.1. de curriculum	Fundamentele Programării, Structuri de date și algoritmi
4.2. de competențe	Abilități medii de programare într-un limbaj de programare de nivel înalt

5. Condiții (acolo unde este cazul)

5.1. de desfășurare a cursului	Sală de curs cu videoproiector
5.2. de desfășurare a seminarului/ laboratorului	Pentru activitatea de laborator este nevoie de calculatoare cu medii de programare avansate, limbajul C++

6.1. Competențele specifice acumulate¹

¹ Se poate opta pentru competențe sau pentru rezultatele învățării, respectiv pentru ambele. În cazul în care se alege o singură variantă, se va șterge tabelul aferent celeilalte opțiuni, iar opțiunea păstrată va fi numerotată cu 6.

Competențe profesionale/esențiale	Descrierea adecvată a paradigmelor de programare și a mecanismelor de limbaj specifice, precum și identificarea diferenței dintre aspectele de ordin semantic și sintactic. Explicarea unor aplicații soft existente, pe niveluri de abstractizare (arhitectură, pachete, clase, metode) utilizând în mod adecvat cunoștințele de bază Elaborarea codurilor sursă adecvate și testarea unitară a unor componente într-un limbaj de programare cunoscut, pe baza unor specificații de proiectare date Testarea unor aplicații pe baza unor planuri de test Dezvoltarea de unități de program și elaborarea documentațiilor aferente
Competențe transversale	Aplicarea regulilor de muncă organizată și eficientă, a unor atitudini responsabile față de domeniul didactic-științific, pentru valorificarea creativă a propriului potențial, cu respectarea principiilor și a normelor de etică profesională Utilizarea unor metode și tehnici eficiente de învățare, informare, cercetare și dezvoltare a capacităților de valorificare a cunoștințelor, de adaptare la cerințele unei societăți dinamice și de comunicare în limba română și într-o limbă de circulație internațională

6.2. Rezultatele învățării

Cunoștințe	Absolventul are abilitatea de a dezvolta, proiecta și crea noi aplicații, sisteme sau produse folosind bunele practici din domeniu. Absolventul are cunoștințe legate de programare, matematică, inginerie și tehnologie și are abilitățile necesare pentru a le folosi în crearea de sisteme informatice complexe.
Aptitudini	Studentul are abilitatea de a înțelege și comunica eficient informațiile. Studentul are abilitatea de a dezvolta, proiecta și crea noi aplicații, sisteme sau produse folosind bunele practici din domeniu Studentul are abilitatea de a aplica reguli generale unor probleme specifice și de a produce soluții relevante.
Responsabilități și autonomie	Studentul are capacitatea de a lucra independent și este capabil să combine informații diverse pentru a formula soluții și genera idei de dezvoltare pentru noi produse și aplicații. Studentul are deprinderile necesare pentru utilizarea instrumentelor de sprijinire a cercetării

7. Obiectivele disciplinei (reieșind din grila competențelor acumulate)

7.1 Obiectivul general al disciplinei	Programarea orientată obiect are drept obiectiv însușirea principiilor de bază a programării cu ajutorul tipurilor abstrakte de date (obiectelor). Să deprindă studentul cu proiectare orientată obiect a problemelor de scară mică/mijlocie și învățarea limbajului de programare C++ și crearea de interfețe grafice utilizator în QT.
--	--

7.2 Obiectivele specifice	După însușirea materialului prezentat la această disciplină studenții ar trebui: • să poată rezolva probleme de dimensiuni mici și medii într-o manieră orientată pe obiecte • să poată evidenția diferența între proiectarea imperativă tradițională și proiectarea orientată pe obiecte • să poată explica structurile de tip clasă ca fiind componente fundamentale, modulare • să înțeleagă rolul moștenirii, polimorfismului, legării dinamice și a structurilor generice în dezvoltarea unor programe reutilizabile • să explice și să folosească diferite strategii de programare referitor la dezvoltarea bazată pe funcționalități, dezvoltarea bazată pe testare, tratarea excepțiilor și utilizarea aserțiunilor formale • să poată scrie programe C++ de dimensiuni mici/medii • în timpul rezolvării unei probleme să folosească clase scrise de alți programatori să înțeleagă și să folosească structurile de date fundamentale: colecții, mulțimi, tabele, liste, stive, cozi, arbori, grafe
----------------------------------	--

8. Conținuturi

8.1 Curs	Metode de predare	Observații
In prima parte a cursului sunt introduse gradat conceptele programarii orientate pe obiecte. Pentru exemplificari este folosit limbajul C++. Sunt supuse dezbaterilor diferite structuri de date ce au fost prezentate in cursul de introducere in informatica. In partea a doua sunt prezentate subiecte mai avansate de programare C++: ierarhii de clase standard, programarea dirijata de evenimente, componente C++ pentru interfata cu utilizatorul, tratarea exceptiilor. Curs 1 Elemente de baza ale limbajului C&C++. - Elemente lexicale. Operatori. Conversii. - Tipuri de date. Variabile. Constante. - Domeniu de vizibilitate si durata de viata a variabilelor. - Spatii de nume. - Instructiuni . - Declararea si definitia functiilor. - Supraincercarea functiilor. Functii inline.	Expunerea Conversația Problematizarea	
Curs 2 Tipuri de date derivate si utilizator si alocare dinamica in C++. - Tipurile tablou si structura. - Tipurile pointer si referinta. - Alocarea si dealocarea memoriei. - Pointeri la functii si pointeri void. Programare modulara in C++. - Fisiere header. Biblioteci. - Implementari modulare de tipuri abstracte de date. - Folosirea tipului pointer void pentru obtinerea genericitatii.	Expunerea Conversația Demonstrația didactică Algoritmizarea	
Curs 3 Metoda programarii orientate-obiect in C++. - Clase si obiecte. - Membrii unei clase. Specificatori de	Expunerea Conversația Demonstrația didactică Algoritmizarea	

- acces. - Constructori / destructori - Diagrame UML pentru clase (membrii, acces).	Problematizarea	
Curs 4 - Supraincercarea operatorilor. - Membrii statici. - Functii Friend.	Expunerea Algoritmizarea Problematizarea	
Curs 5 - Clase: containere si iteratori - o Lista simplu inlantuita si lista dublu inlantuita - o Stiva, Coadă, Tabele de dispersie - o Relatia de asociere/agregare intre clase - reprezentare UML	Expunerea Algoritmizarea Problematizarea	
Curs 6 - Relatia de mostenire - o Mostenire simpla. Clase derivate. - o Principiul substitutiei. - o Suprascrierea metodelor. - o Mostenire multipla. - o Relatia de specializare/generalizare intre clase - reprezentare UML.	Expunerea Conversația Algoritmizarea Problematizarea	
Curs 7 - Clase abstracte si Polimorfism. - o Metode virtuale. - o Legare dinamica. - o Mostenire virtuala. - o Reutilizare cod (mostenire/compozitie). - o Conversii (upcast/downcast).	Expunerea Demonstrația didactică Algoritmizarea Problematizarea	
Curs 8 - Proiectare orientata-obiect si proiectare bazata pe interfete. - o Clase abstracte, interfete. - o Reprezentare UML pentru interfete. - o Proiectarea orientata-obiect a unei biblioteci de structuri de date.	Expunerea Conversația Demonstrația didactică Algoritmizarea Problematizarea	
Curs 9 - Sabloane (Template). - o Functii template. Clase template. - o Reutilizarea codului sursa.	Expunerea Conversația Algoritmizarea Problematizarea	
Curs 10 - Operatii de intrare/iesire. - o Fluxuri de intrare/ iesire. - o Ierarhii de clase pentru I/O. - o Formatare. Manipulatori. - o Lucrul cu fisiere.	Expunerea Conversația Demonstrația didactică Algoritmizarea Problematizarea	
Curs 11 - Tratarea exceptiilor. - o Notiunea de exceptie in programare - o Tratarea exceptiilor in C++.	Expunerea Conversația Demonstrația didactică Algoritmizarea Problematizarea	
Curs 12 - Biblioteca STL - o Clase container si iterator.	Expunerea Conversația Demonstrația didactică	

○ Folosirea algoritmilor din STL.	Algoritmizarea Problematizarea	
Curs 13 - Elemente de programare bazată pe evenimente si Interfete grafice ○ MCF ○ VCL ○ QT	Expunerea Conversația Demonstrația didactică Algoritmizarea Problematizarea	
Curs 14 - Sabloane de proiectare creaționale, structurale, comportamentale.	Expunerea Conversația Demonstrația didactică Algoritmizarea Problematizarea	

Bibliografie

1. A.V. Aho, J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Addison-Wesley Publ., Massachusetts, 1983.
2. R. Andonie, I. Garbacea, Algoritmi fundamentali. O perspectiva C++, Editura Libris,
3. Alexandrescu, Programarea moderna in C++. Programare generica si modele de proiectare aplicate, Editura Teora, 2002
4. M. Frentiu, B. Parv, Elaborarea programelor. Metode si tehnici moderne, Ed. Promedia, Cluj-Napoca, 1994.
5. E. Horowitz, S. Sahni, D. Mehta, Fundamentals of Data Structures in C++, Computer Science Press, Oxford, 1995.
6. K.A. Lambert, D.W. Nance, T.L. Naps, Introduction to Computer Science with C++, West Publishing Co., New-York, 1996.
7. L. Negrescu, Limbajul C++, Ed. Albastra, Cluj-Napoca 1996.
8. Dan Roman, Ingineria programarii obiectuale, Editura Albastra, Cluj_Napoca, 1996.
9. B. Stroustup, The C++ Programming Language, Addison Wesley, 1998.
10. Bruce Eckel, Thinking in C++, www.bruceeckel.com

8.2 Seminar / laborator	Metode de predare	Observații
S 1. Probleme simple in C++. Functii. Variabile (locale si globale) si vizibilitatea lor. Vectori (uni si multidimensionali) si structuri. L 1. Strategii și metode inteligente de rezolvare a jocurilor	Conversația Algoritmizarea Descoperirea Studiul individual Exercițiul	Fiecare seminar dureaza 2 ore si se desfasoara o data la 2 saptamani Fiecare laborator dureaza 2 ore si se desfasoara o data la 2 saptamani
S 2. TAD container cu elemente generice (void*): reprezentare vizibila si ascunsa. Citiri si scrieri din/in fisiere. L 2. Specificarea, proiectarea si implementarea unor probleme simple in C/C++. Aspecte generale ale limbajelor C si C++. Versiuni structurate si modulare ale aplicatiilor C/C++. Specificarea, proiectarea si implementarea unui TAD in C/C++. Reprezentarea unui TAD. Operatiile unui TAD. Utilizarea unui TAD.	Conversația Algoritmizarea Problematizarea Studiul de caz Cooperarea Studiul individual Exercițiul	
S 3. Clase. Clase simple. Supraincaracrea operatorilor. Clase cu date membre de tip obiect. L 3. Specificarea, proiectarea si implementarea unui TAD container in C/C++. Utilizarea elementelor generice pentru popularea containerului.	Conversația Algoritmizarea Problematizarea Descoperirea Simularea Studiul individual Exercițiul	

Reprezentarea TAD vector cu elemente generice. Operatiile TAD vector cu elemente generice. Utilizarea TAD vector cu elemente generice. Specificarea, proiectarea si implementarea unui TAD container dinamic in C/C++. Utilizarea elementelor generice si a iteratorilor. Reprezentarea TAD container dinamic cu elemente generice. Operatiile TAD container cu elemente generice. Utilizarea TAD container cu elemente generice. Utilizarea iteratorilor.		
S 4. Clase de tip lista dinamica si iteratori. Mostenire. L 4. Clase si obiecte. Specificarea, proiectarea si implementarea unei clase in C++. Constructori, destructor, accesorii, mutatori. Supraincarcarea operatorilor. Obiecte statice si dinamice. Clase cu date membre de tip obiect. Specificarea, proiectarea si implementarea acestor clase. Obiecte dinamice.	Conversația Algoritmizarea Problematizarea Studiul de caz Brainstorming-ul Studiul individual Exercițiul	
S 5. Clase abstracte si interfete. Polimorfism. L 5. Clase si obiecte. Containere dinamice cu obiecte (liste dinamice, dictionare, tabele de dispersie). Iteratori pentru aceste containere. Specificarea, proiectarea si implementarea unei clase in C++. Constructori, destructor, accesorii, mutatori. Clase cu date membre de tip obiect. Obiecte dinamice. Supraincarcarea operatorilor. Mostenire.	Conversația Algoritmizarea Problematizarea Descoperirea Studiul de caz Studiul individual Exercițiul	
S 6. Sabloane si exceptii. L 6. Clase si obiecte. Mostenire. Polimorfism. Specificarea, proiectarea si implementarea claselor derivate. Clase abstracte si interfete. Sabloane. Sabloane si exceptii. Specificarea, proiectarea si implementarea claselor in C++. Implementarea unei probleme respectand o diagrama UML. Sabloane de proiectare.	Conversația Algoritmizarea Studiul de caz Simularea Studiul individual Exercițiul	
S 7. Probleme complexe implementate pe baza unei diagrame UML. Sabloane de proiectare. L 7. -	Conversația Algoritmizarea Problematizarea Studiul de caz Brainstorming-ul Studiul individual Exercițiul	
Bibliografie 1. A.V. Aho, J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Addison-Wesley Publ., Massachusetts, 1983. 2. R. Andonie, I. Garbacea, Algoritmi fundamentali. O perspectiva C++, Editura Libris, 3. Alexandrescu, Programarea moderna in C++. Programare generica si modele de proiectare aplicate, Editura Teora, 2002		

4. M. Frentiu, B. Parv, Elaborarea programelor. Metode si tehnici moderne, Ed. Promedia, Cluj-Napoca, 1994.
5. E. Horowitz, S. Sahni, D. Mehta, Fundamentals of Data Structures in C++, Computer Science Press, Oxford, 1995.
6. K.A. Lambert, D.W. Nance, T.L. Naps, Introduction to Computer Science with C++, West Publishing Co., New-York, 1996.
7. L. Negrescu, Limbajul C++, Ed. Albastra, Cluj-Napoca 1996.
8. Dan Roman, Ingineria programarii obiectuale, Editura Albastra, Cluj_Napoca, 1996.
9. B. Stroustup, The C++ Programming Language, Addison Wesley, 1998.
10. Bruce Eckel, Thinking in C++, www.bruceeckel.com
11. L. Negrescu, Limbajul C++, Ed. Albastra, Cluj-Napoca 1996.

9. Coroborarea conținuturilor disciplinei cu așteptările reprezentanților comunității epistemice, asociațiilor profesionale și angajatori reprezentativi din domeniul aferent programului

- Cursul respecta recomandările curriculare IEEE si ACM pentru studiile in informatica
- Cursul exista in programa de studiu a numeroase facultatilor de profil din intreaga lume
- Companiile de software considera continutul cursului ca fiind util in dezvoltarea abilitatilor de modelare si programare ale studentilor

10. Evaluare

Tip activitate	10.1 Criterii de evaluare	10.2 Metode de evaluare	10.3 Pondere din nota finală
10.4 Curs	Cunoasterea conceptelor de baza ale domeniului Aplicarea principiilor de programare orientata obiect din continutul cursului pentru rezolvarea problemelor complexe si dificile	Examen scris	30%
	Specificarea, proiectarea, implementarea si testarea aplicatiilor Rezolvarea efectiva a problemelor cu ajutorul metodelor anterior implementate	Observarea sistematică a studentului de-a lungul semestrului in rezolvarea problemelor de seminar Observarea sistematică a studentului de-a lungul semestrului in rezolvarea problemelor de laborator	10% 30%
10.5 Seminar/laborator	Aplicarea principiilor de programare orientata obiect din continutul cursului pentru rezolvarea problemelor complexe si dificile	Proiect practic	30%
10.6 Standard minim de performanță			
- Fiecare student trebuie sa demonstreze ca a atins un nivel acceptabil de cunoastere si intelegere a domeniului, ca este capabil sa exprime cunostintele intr-o forma coerenta, ca are capacitatea de a stabili anumite conexiuni si de a utiliza cunostintele in rezolvarea unor probleme. - Pentru a putea promova examenul studentul trebuie: - Sa fie prezent la cel putin 6 seminarii si 12 laboratoare laboratoare. Studenții care nu au prezență la minim 6			

- seminarii si la minim 12 laboratoare nu se pot prezenta la examen nici în sesiunea de restanțe
- Sa obtina pe fiecare activitate minim nota 5 și nota finală să fie minim 5

11. Etichete ODD (Obiective de Dezvoltare Durabilă / Sustainable Development Goals)²

Nu se aplică.

Data completării:

13 noiembrie 2025

Semnătura titularului de curs

Prof. dr. Laura Diosan

Semnătura titularului de seminar

Prof. dr. Laura Diosan

Data avizării în departament:

...

Semnătura directorului de departament

Conf.dr. Adrian STERCA

² Păstrați doar etichetele care, în conformitate cu [Procedura de aplicare a etichetelor ODD în procesul academic](#), se potrivesc disciplinei și ștergeți-le pe celelalte, inclusiv eticheta generală pentru *Dezvoltare durabilă* - dacă nu se aplică. Dacă nicio etichetă nu descrie disciplina, ștergeți-le pe toate și scrieți "*Nu se aplică.*".