FIŞA DISCIPLINEI

Bazele programării orientate obiect
Anul universitar 2025-2026

1. Date despre program

1.1. Instituţia de învăţământ superior	Universitatea Babeş-Bolyai Cluj-Napoca
1.2. Facultatea	Facultatea de Matematică şi Informatică
1.3. Departamentul	Departamentul de Informatică
1.4. Domeniul de studii	Matematică / Informatică
1.5. Ciclul de studii	Licenţă
1.6. Programul de studii / Calificarea	Matematică - Informatică
1.7. Forma de învăţământ	Cu frecvenţă

2. Date despre disciplină

2.1. Denumirea disciplinei	Bazele programării orientate obiect					Codul disciplinei	MLR5234
2.2. Titularul activităţilor de curs	Prof. Dr. Dioşan Laura						
2.3. Titularul activităţilor de seminar	Prof. Dr. Dioşan 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 activităţilor 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 învăţământ	70	din care: 3.5. curs	28	3.6 seminar/laborator/proiect	**42**
Distribuţia fondului de timp pentru studiul individual (SI) şi activităţi de autoinstruire (AI)					**ore**
Studiul după manual, suport de curs, bibliografie şi notiţe (AI)					30
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate şi pe teren					25
Pregătire seminare/ laboratoare/ proiecte, teme, referate, portofolii şi 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 activităţi [de ex.: comunicare bidirecţională cu titularul de disciplină / tutorele]					
3.7. Total ore studiu individual (SI) şi activităţi de autoinstruire (AI)				**80**	
3.8. Total ore pe semestru				**150**	
3.9. Numărul de credite				**6**	

4. Precondiţii (acolo unde este cazul)

4.1. de curriculum	Fundamentele Programării, Structuri de date şi algoritmi
4.2. de competenţe	Abilităţi medii de programare într-un limbaj de programare de nivel înalt

5. Condiţii (acolo unde este cazul)

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

6.1. Competenţele specifice acumulate[1]

[1] Se poate opta pentru competenţe sau pentru rezultatele învăţării, respectiv pentru ambele. În cazul în care se alege o singură variantă, se va şterge tabelul aferent celeilalte opţiuni, iar opţiunea păstrată va fi numerotată cu 6.

Competențe profesionale/esențiale	Descrierea adecvată a paradigmelor de programare şi a mecanismelor de limbaj specifice, precum şi identificarea diferenţei dintre aspectele de ordin semantic şi sintactic. Explicarea unor aplicaţii soft existente, pe niveluri de abstractizare (arhitectură, pachete, clase, metode) utilizând in mod adecvat cunoştinţele de bază Elaborarea codurilor sursă adecvate şi testarea unitară a unor componente într-un limbaj de programare cunoscut, pe baza unor specificaţii de proiectare date Testarea unor aplicaţii pe baza unor planuri de test Dezvoltarea de unităţi de program şi elaborarea documentaţiilor aferente
Competențe transversale	Aplicarea regulilor de muncă organizată şi eficientă, a unor atitudini responsabile faţă de domeniul didactic-ştiinţific, pentru valorificarea creativă a propriului potenţial, cu respectarea principiilor şi a normelor de etică profesională Utilizarea unor metode şi tehnici eficiente de învăţare, informare, cercetare şi dezvoltare a capacităţilor de valorificare a cunoştinţelor, de adaptare la cerinţele unei societăţi dinamice şi de comunicare în limba română şi într-o limbă de circulaţie internaţională

6.2. Rezultatele învăţării

Cunoștințe	Absolventul are abilitatea de a dezvolta, proiecta şi crea noi aplicaţii, sisteme sau produse folosind bunele practici din domeniu. Absolventul are cunoştinţe legate de programare, matematică, inginerie şi tehnologie si are abilităţile necesare pentru a le folosi in crearea de sisteme informatice complexe.
Aptitudini	Studentul are abilitatea de a înţelege şi comunica eficient informaţiile. Studentul are abilitatea de a dezvolta, proiecta şi crea noi aplicaţii, sisteme sau produse folosind bunele practici din domeniu Studentul are abilitatea de a aplica reguli generale unor probleme specifice şi de a produce soluţii relevante.
Responsabilități și autonomie	Studentul are capacitatea de a lucra independent şi este este capabil să combine informaţii diverse pentru a formula soluţii şi genera idei de dezvoltare pentru noi produse şi aplicaţii. Studentul are deprinderile necesare pentru utilizarea instrumentelor de sprijinire a cercetării

7. Obiectivele disciplinei (reieşind din grila competenţelor acumulate)

7.1 Obiectivul general al disciplinei	• Programarea orientata obiect are drept obiectiv însuşirea princiilor de bază a programarii cu ajutorul tipurilor abstrate de date (obiectelor). Să deprindă studentul cu proiectare orientată obiect a problemelor de scară mică/mijlocie şi învăţarea limbajului de programare C++ şi crearea de interfeţe 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 şi medii într-o manieră orientată pe obiecte • să poată evidenția diferența între proiectarea imperativă tradiţională şi proiectarea orientată pe obiecte • să poată explica structurile de tip clasă ca fiind componente fundamentale, modulare • să înţeleagă rolul moştenirii, polimorfismului, legării dinamice şi a structurilor generice în dezvoltarea unor programe reutilizabile • să explice şi să folosească diferite strategii de programare referitor la dezvoltarea bazată pe funcționalități, dezvoltarea bazată pe testare, tratarea excepţiilor şi utilizarea aserţiunilor formale • să poată scrie programe C++ de dimensiuni mici/medii • în timpul rezolvării unei probleme să folosească clase scrise de alți programatori să înţeleagă şi să folosească structurile de date fundamentale: colecţii, mulţimi, tabele, liste, stive, cozi, arbori, grafe

8. Conţinuturi

8.1 Curs	Metode de predare	Observa ţii
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. - Supraincarcarea functiilor. Functii inline.	Expunerea Conversaţia 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 Conversaţia Demonstraţia didactică Algoritmizarea	
Curs 3 Metoda programarii orientate-obiect in C++. - Clase si obiecte. - Membrii unei clase. Specificatori de	Expunerea Conversaţia Demonstraţia didactică Algoritmizarea	

acces. - Constructori / destructori - Diagrame UML pentru clase (membrii, acces).	Problematizarea	
Curs 4 - Supraincarcarea operatorilor. - Membrii statici. - Functii Friend.	Expunerea Algoritmizarea Problematizarea	
Curs 5 - Clase: containere si iteratori o Lista simplu inlantuita si lista dublu inlantuita o Stiva, Coada, 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 Conversaţia 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 Demonstraţia 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 Conversaţia Demonstraţia didactică Algoritmizarea Problematizarea	
Curs 9 - Sabloane (Template). o Functii template. Clase template. o Reutilizarea codului sursa.	Expunerea Conversaţia Algoritmizarea Problematizarea	
Curs 10 - Operatii de intrare/iesire. o Fluxuri de intrare/ iesire. Ierarhii de clase pentru I/O. o Formatare. Manipulatori. o Lucrul cu fisiere.	Expunerea Conversaţia Demonstraţia didactică Algoritmizarea Problematizarea	
Curs 11 - Tratarea exceptiilor. o Notiunea de exceptie in programare o Tratarea exceptiilor in C++.	Expunerea Conversaţia Demonstraţia didactică Algoritmizarea Problematizarea	
Curs 12 - Biblioteca STL o Clase container si iterator.	Expunerea Conversaţia Demonstraţia didactică	

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

Bibliografie

1. A.V. Aho, J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Addisson-Wessley 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	Observa ţii
S 1. Probleme simple in C++. Functii. Variabile (locale si globale) si vizibilitatea lor. Vectori (uni si multidimensionali) si structuri. L 1. Strategii şi metode inteligente de rezolvare a jocurilor	Conversaţia Algoritmizarea Descoperirea Studiul individual Exerciţiul	Fiecare seminar dureaza 2 ore si se desfasoa ra o data la 2 sapta mani Fiecare laboratordureaza 2 ore si se desfasoa ra o data la 2 sapta mani
S 2. TAD container cu elemente generice (void*): representare 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.	Conversaţia Algoritmizarea Problematizarea Studiul de caz Cooperarea Studiul individual Exerciţiul	
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.	Conversaţia Algoritmizarea Problematizarea Descoperirea Simularea Studiul individual Exerciţiul	

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, accesori, 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, accesori, mutatori. Clase cu date membre de tip obiect. Obiecte dinamice. Supraincaracarea 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 proeictare. 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, Addisson-Wessley 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 conţinuturilor disciplinei cu aşteptările reprezentanţilor comunităţii epistemice, asociaţiilor profesionale şi angajatori reprezentativi din domeniul aferent programului

- Cursul respecta recomandarile curicullare 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 principilor 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 principilor de programare orientata obiect din continutul cursului pentru rezolvarea problemelor complexe si dificile	Proiect practic	30%

10.6 Standard minim de performanţă

- 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. Studenţii care nu au prezenţă la minim 6

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

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

Nu se aplică.

Data completării:	Semnătura titularului de curs	Semnătura titularului de seminar
13 noiembrie 2025	Prof. dr. Laura Diosan	Prof. dr. Laura Diosan

Data avizării în departament:	Semnătura directorului de departament
...	Conf.dr. Adrian STERCA

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