

FIȘA DISCIPLINEI

Programare orientată obiect

Anul universitar 2026-2027

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	Informatică
1.5. Ciclul de studii	Licență
1.6. Programul de studii / Calificarea	Informatică engleză
1.7. Forma de învățământ	Cu frecvență

2. Date despre disciplină

2.1. Denumirea disciplinei	Programare orientată obiect / Object oriented programming			Codul disciplinei	MLE5006
2.2. Titularul activităților de curs	Conf. dr. Bocicor Maria Iuliana				
2.3. Titularul activităților de seminar	Conf. dr. Bocicor Maria Iuliana				
2.4. Anul de studiu	1	2.5. Semestrul	2	2.6. Tipul de evaluare	Examen
2.7. Regimul disciplinei	Obligatoriu		2.8. Tipul disciplinei	Disciplină de specializare (DS)	

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	1 sem 2 lab
3.4. Total ore din planul de învățământ	70	din care: 3.5. curs	28	3.6 seminar/laborator	14+28
Distribuția fondului de timp pentru studiul individual (SI) și activități de autoinstruire (AI)					ore
Studiul după manual, suport de curs, bibliografie și notițe (AI)					20
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren					5
Pregătire seminare/ laboratoare/ proiecte, teme, referate, portofolii și eseuri					19
Tutoriat (consiliere profesională)					4
Examinări					7
Alte activități					
3.7. Total ore studiu individual (SI) și activități de autoinstruire (AI)				55	
3.8. Total ore pe semestru				125	
3.9. Numărul de credite				5	

4. Precondiții (acolo unde este cazul)

4.1. de curriculum	Fundamentele Programării
4.2. de competențe	Cunoștințe 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	Sală de laborator cu calculatoare dotate cu limbajul de programare C++ și Qt

6.1. Competențele dobândite în urma absolvirii programului de studii (se preiau din planul de învățământ)¹

¹ Se vor prelua din Planul de învățământ al programului de studii acele competențe profesionale și/sau transversale la dezvoltarea cărora contribuie disciplina pentru care se elaborează fișa disciplinei. Pentru fiecare competență se va prelua întregul enunț, inclusiv codul competenței, cu formularea care apare în planul de

Competențe profesionale	
Codul competenței	Competență
CP1	Creează softuri
CP4	Definește arhitectura software
CP5	Definește cerințe tehnice
Competențe transversale	
Codul competenței	Competență
CT2	Solve problems
CT3	Think analytically

6.2. Rezultatele învățării specifice programului de studii (se preiau din planul de învățământ)²

Rezultatele învățării vizate prin disciplină		
Codul competenței	Cunoștințe și înțelegere (Knowledge and understanding)	Abilități academice specifice (Specific academic skills)
CP1	Studentul/absolventul identifică, explică și argumentează concepte fundamentale de structuri de date, algoritmi și paradigme de programare, precum și a arhitecturii calculatoarelor.	Studentul/absolventul elaborează, dezvoltă și demonstrează soluții software complexe utilizând algoritmi eficienți și paradigme diverse de programare.
CP4	Studentul/absolventul cunoaște și explică paradigmele moderne de programare, arhitecturi software și metodologii de dezvoltarea proiectelor software.	Studentul/absolventul proiectează, planifică, construiește, dezvoltă aplicații software scalabile și utilizează eficient resursele hardware și software.
CP5	Studentul/absolventul alege, explică și specifică fundamentele matematice aplicate în informatică, inclusiv logica formală, algebra, probabilitățile și statisticele.	Studentul/absolventul aplică, evaluează, propune metodele matematice pentru modelarea, simularea și rezolvarea problemelor informatice.
CT2, CT3	Studentul/absolventul are cunoștințele necesare pentru a înțelege și soluționa probleme complexe, pentru a planifica și organiza procese avansate în diverse domenii.	Absolventul este capabil să identifice probleme complexe și să examineze probleme conexe pentru a dezvolta opțiuni de rezolvare și implementa soluții. Absolventul are abilitatea de a aplica reguli generale unor probleme specifice și de a produce soluții relevante. Absolventul este capabil să combine informații diverse pentru a formula soluții și genera idei de dezvoltare pentru noi produse și aplicații.

7. Rezultatele învățării specifice disciplinei

Cunoștințe și înțelegere (Knowledge and understanding)
1. Studentul are cunoștințe necesare pentru dezvoltarea programelor și aplicațiilor software folosind programarea orientată obiect și limbajul de programare C++ și pentru crearea interfețelor grafice folosind Qt.
2. Studentul are abilitatea de a dezvolta, proiecta și crea noi aplicații folosind bunele practici din domeniu.
Abilități academice specifice (Specific academic skills)

învățământ, fără modificări. Dacă nu se preia nici o competență din oricare din cele două categorii, se șterge linia din tabel aferentă acelei categorii.

² Se menționează rezultatele învățării specifice programului de studiu la dezvoltarea cărora contribuie disciplina pentru care se elaborează fișa. Enunțurile, preluate fără modificări din Planul de învățământ în funcție de tipul disciplinei (DF/DS/DC) se trec în dreptul competenței asociate.

1. Studentul este capabil să înțeleagă și să utilizeze concepte ale programării orientate obiect pentru a dezvolta aplicații software de complexitate medie.
2. Studentul este capabil să distingă între paradigmele de programare imperativă tradițională și cea orientată pe obiect, să explice rolul claselor ca blocuri modulare fundamentale și să demonstreze înțelegerea conceptelor de moștenire, polimorfism, legare dinamică și structuri generice în crearea de cod reutilizabil și ușor de întreținut.
3. Absolventul are abilitatea de a aplica reguli generale unor probleme specifice și de a produce soluții relevante.

8. Conținuturi

8.1 Curs	Metode de predare	Observații
<p>1. Elemente de bază ale limbajului C.</p> <ul style="list-style-type: none"> • Elemente de bază ale limbajului C. • Elemente lexicale. Operatori. Conversii. • Tipuri de date. Variabile. Constante. • Domeniul de vizibilitate și durata de viață a variabilelor. <p>Declararea și definirea funcțiilor. Supraîncărcarea funcțiilor. Funcții inline.</p>		
<p>2. Programare modulară în C/C++.</p> <ul style="list-style-type: none"> • Funcții. Parametri. • Pointeri și gestiunea memoriei. • Pointeri la funcții • Fișiere header. Biblioteci. <p>Implementarea modulară a TAD-urilor.</p>		
<p>3. Programare orientată obiect în C++.</p> <ul style="list-style-type: none"> • Clase și obiecte. • Definirea claselor. • Crearea și distrugerea obiectelor. • Supraincercarea operatorilor. <p>Elemente statice și prietene.</p>		
<p>4. Elemente de programare generică</p> <ul style="list-style-type: none"> • Funcții/clase parametrizate. Mecanismul de template din C++. • Containere și iteratori – biblioteca STL. • Algoritmi STL. 	<ul style="list-style-type: none"> • Expunerea interactivă • Explicația • Conversația • Exemple • Demonstrația didactică 	
<p>5. Moștenire</p> <ul style="list-style-type: none"> • Moștenire simplă. Clase derivate. • Funcții speciale în clase relativ la moștenire. • Principiul substituției. • Supraîncărcarea metodelor. • Moștenire multiplă. • Relații de specializare/generalizare – reprezentări UML. 		
<p>6. Polimorfism</p> <ul style="list-style-type: none"> • Moștenire, polimorfism. • Legare statică și dinamică. • Metode virtuale. • Casting. • Clase abstracte. 		
<p>7. Stream-uri și gestiunea excepțiilor</p> <ul style="list-style-type: none"> • Stream-uri input/output. • Operatori de inserție, extracție. 		

<ul style="list-style-type: none"> • Formatare, manipulators, flags. • Fișiere text. • Gestiunea excepțiilor. Cod exception-safe. 		
8. Gestiunea resurselor și RAII <ul style="list-style-type: none"> • Resource Acquisition Is Initialization (RAII) • Smart pointers • RAII în STL. Smart pointers în STL. 		
9. Interfețe grafice utilizator (GUI) <ul style="list-style-type: none"> • QT Toolkit: instalare, instrumente și module Qt • Componente grafice utilizator • Layout management • Qt Designer 		
10. Elemente de programare bazată pe evenimente <ul style="list-style-type: none"> • Callbacks. • Evenimente: Semnale și sloturi Qt • Proiectare GUI 		
11. Elemente de programare bazată pe evenimente <ul style="list-style-type: none"> • Șablonul MVC. • Componente grafice cu modele • Utilizarea modelelor predefinite. Implementarea modelelor personalizate. • Studiu de caz: aplicație pentru gestiunea genelor. 		
12. Șabloane de proiectare <ul style="list-style-type: none"> • Șabloane de proiectare creaționale, structurale, comportamentale. • Exemple 		
13. Șabloane de proiectare <ul style="list-style-type: none"> • Șablonul de proiectare Adapter. • Șablonul de proiectare Observer. • Șablonul de proiectare Iterator. • Șablonul de proiectare Composite. • Șablonul de proiectare Strategy. • Studii de caz și exemple. 		
14. Recapitulare <ul style="list-style-type: none"> • Recapitularea celor mai importante teme discutate la curs. • Ghid de examinare. 		
Bibliografie		
<ol style="list-style-type: none"> 1. B. Stroustrup. The C++ Programming Language, Addison Wesley, 1998. 2. Bruce Eckel. Thinking in C++, Prentice Hall, 1995. 3. A. Alexandrescu. Programarea moderna in C++: Programare generica si modele de proiectare aplicate, Editura Teora, 2002. 4. S. Meyers. Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition), Addison-Wesley, 2005. 5. S. Meyers. More effective C++: 35 New Ways to Improve Your Programs and Designs, Addison-Wesley, 1995. 6. B. Stroustrup. A Tour of C++, Addison Wesley, 2013. 7. C++ reference (http://en.cppreference.com/w/). 8. Qt Documentation (http://doc.qt.io/qt-6/). 9. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Longman Publishing, 1995. 		
8.2 Seminar	Metode de predare	Observații

1. Probleme simple în C. Funcții. Structuri și vectori.	<ul style="list-style-type: none"> • Expunerea interactivă • Explicația • Conversația • Exemple • Demonstrația didactică 	Seminarul este structurat sub forma a 2 ore din 2 în 2 săptămâni.
2. Programare modulară.		
3. Clase. Supraîncărcarea operatorilor. Clase cu obiecte ca date membri. Templates (vector dinamic).		
4. Moștenire, polimorfism		
5. Fișiere, excepții, containere STL, iteratori, algoritmi.		
6. Interfete grafice utilizator		
7. Probleme complexe implementate pe baza diagramelor UML. Șabloane de proiectare.		
Bibliografie		
<ol style="list-style-type: none"> 1. B. Stroustrup. The C++ Programming Language, Addison Wesley, 1998. 2. Bruce Eckel. Thinking in C++, Prentice Hall, 1995. 3. A. Alexandrescu. Programarea moderna in C++: Programare generica si modele de proiectare aplicate, Editura Teora, 2002. 4. S. Meyers. Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition), Addison-Wesley, 2005. 5. S. Meyers. More effective C++: 35 New Ways to Improve Your Programs and Designs, Addison-Wesley, 1995. 6. B. Stroustrup. A Tour of C++, Addison Wesley, 2013. 7. C++ reference (http://en.cppreference.com/w/). 8. Qt Documentation (http://doc.qt.io/qt-6/). 9. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Longman Publishing, 1995. 		
8.3 Laborator	Metode de predare	Observații
1. Instalare compilator C++ (MSVC/MinGW) și IDE (Visual Studio/Eclipse CDT). Aspecte generale C/C++.	<ul style="list-style-type: none"> • Explicația • Conversația 	
2. Programare simple în C.		
3. Dezvoltare software bazată pe funcționalități. Arhitectură stratificată. Dezvoltare bazată pe teste. Programare modulară. (I)		
4. Dezvoltare software bazată pe funcționalități. Arhitectură stratificată. Dezvoltare bazată pe teste. Programare modulară. (II)		
5. Programare orientată obiect în C++. (I)		
6. Programare orientată obiect în C++. (II)		
7. Test de laborator.		
8. Moștenire și polimorfism.		
9. Fișiere text, excepții. Containere STL, iteratori și algoritmi STL.		
10. Test de laborator.		
11. GUI folosind Qt. (I)		
12. GUI folosind Qt. (II)		
13. Test de laborator.		
14. Predare laboratoare.		
Bibliografie		
<ol style="list-style-type: none"> 1. B. Stroustrup. The C++ Programming Language, Addison Wesley, 1998. 2. Bruce Eckel. Thinking in C++, Prentice Hall, 1995. 3. A. Alexandrescu. Programarea moderna in C++: Programare generica si modele de proiectare aplicate, Editura Teora, 2002. 4. S. Meyers. Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition), Addison-Wesley, 2005. 5. S. Meyers. More effective C++: 35 New Ways to Improve Your Programs and Designs, Addison-Wesley, 1995. 6. B. Stroustrup. A Tour of C++, Addison Wesley, 2013. 		

7. C++ reference (<http://en.cppreference.com/w/>).
8. Qt Documentation (<http://doc.qt.io/qt-6/>).
9. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Longman Publishing, 1995.

9. Evaluare

Tip activitate	9.1 Criterii de evaluare ³	9.2 Metode de evaluare ⁴	9.3 Pondere din nota finală
9.4 Curs	Corectitudinea și completitudinea cunoștințelor acumulate și capacitatea de a proiecta și implementa programe C++.	Examen scris (în sesiune)	40%
9.5 Seminar/laborator	Capacitatea de a proiecta, testa și depana programe C++ folosind Qt.	Evaluare practică (în sesiune)	20%
	Corectitudinea temelor de laborator și teste de laborator.	Programe, documentații. Observarea continuă pe parcursul semestrului. Teste de laborator.	40%
9.6 Standard minim de promovare			
<ul style="list-style-type: none"> • Fiecare student trebuie să demonstreze că a atins un nivel acceptabil de cunoaștere și înțelegere a domeniului, că este capabil să exprime cunoștințele într-o formă coerentă, că are capacitatea de a stabili anumite conexiuni și de a utiliza cunoștințele în rezolvarea unor probleme în limbajul de programare C++. • Pentru promovare, este obligatorie prezența la minim 5 seminare și 12 laboratoare . • Pentru promovare este ca necesar notele pentru toate evaluările (activitate de laborator, evaluare practica, examen scris) sa fie minim 5 și nota finală să fie minim 5. 			

10. Etichete ODD (Obiective de Dezvoltare Durabilă / Sustainable Development Goals)⁵

	Eticheta generală pentru Dezvoltare durabilă							
								
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

³ Criteriile de evaluare trebuie să reflecte direct rezultatele învățării vizate la nivel de program de studii, respectiv la nivel de disciplină. Mai concret, se evaluează achizițiile de învățare menționate în rezultatele anticipate ale învățării.

⁴ Se recomandă stabilirea atât a metodelor de evaluare finală, cât și a strategiei de evaluare pe parcurs.

⁵ Selectați o singură etichetă, cea care, în conformitate cu [Procedura de aplicare a etichetelor ODD în procesul academic](#), se potrivește cel mai bine disciplinei. Dacă disciplina tratează tema dezvoltării durabile la modul general (de ex. prin prezentarea/introducerea cadrului general al dezvoltării durabile etc.) atunci se poate alocă eticheta generală de Dezvoltare Durabilă. Dacă niciuna dintre etichete nu descrie disciplina, selectați ultima opțiune: „Nu se aplică nici o etichetă”.

								Nu se aplică nici o etichetă
								

Data completării:

...

Semnătura titularului de curs

Conf. Dr. Bocicor Maria Iuliana

Semnătura titularului de seminar

Conf. Dr. Bocicor Maria Iuliana

Data avizării în departament:

...

Semnătura directorului de departament

Conf.dr. Adrian STERCA