FIŞA DISCIPLINEI

Teoria automatelor si compilatoarelor

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	**Informatică**
1.5. Ciclul de studii	**Licenţă**
1.6. Programul de studii / Calificarea	**Inteligenta Artificiala**
1.7. Forma de învăţământ	Cu frecventa

2. Date despre disciplină

2.1. Denumirea disciplinei	**Teoria automatelor si compilatoarelor**					Codul disciplinei	**MLE5206**
2.2. Titularul activităţilor de curs	Prof.dr. Simona Motogna						
2.3. Titularul activităţilor de seminar	Prof.dr. Simona Motogna						
2.4. Anul de studiu	3	2.5. Semestrul	5	2.6. Tipul de evaluare	E	2.7. Regimul disciplinei	obligatorie

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

3.1. Număr de ore pe săptămână	**6**	din care: 3.2. curs	**2**	3.3. seminar/ laborator/proiect	2+2
3.4. Total ore din planul de învăţământ	84	din care: 3.5. curs	28	3.6 seminar/laborator/proiect	**56**
Distribuţia fondului de timp pentru studiul individual (SI) şi activităţi de autoinstruire (AI)					**Ore**
Studiul după manual, suport de curs, bibliografie şi notiţe (AI)					10
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate şi pe teren					5
Pregătire seminare/ laboratoare/ proiecte, teme, referate, portofolii şi eseuri					10
Tutoriat (consiliere profesională)					6
Examinări					10
Alte activităţi					-
3.7. Total ore studiu individual (SI) şi activităţi de autoinstruire (AI)				**41**	
3.8. Total ore pe semestru				**125**	
3.9. Numărul de credite				**5**	

4. Precondiţii (acolo unde este cazul)

4.1. de curriculum	Fundamentele programarii, Structuri de date si algoritmi
4.2. de competenţe	Abilitati medii de programare intr-un limbaje de nivel inalt

5. Condiţii (acolo unde este cazul)

5.1. de desfăşurare a cursului	Sala de curs cu proiector
5.2. de desfăşurare a seminarului/ laboratorului	Laborator cu calculatoare Software licentiat (.NET, Java, Python etc.)

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	• dezvoltă prototipul pentru software • proiecteaza sistemul informatic • remediaza erorile din software
Competențe transversale	• lucrează în echipe • gândește analitic

6.2. Rezultatele învăţării

Cunoştinţe	- Absolventul cunoaşte şi înţelege fundamentele matematice necesare dezvoltării algoritmilor inteligenţi şi este capabil să le utilizeze pentru implementarea acestor algoritmi. - Absolventul are cunoştinţe legate de programare, matematică şi tehnologie si are abilităţile necesare pentru a le folosi in crearea de sisteme informatice complexe.
Aptitudini	- Absolventul este capabil să evalueze, atât în mod cantitativ şi cât şi în mod calitativ, performanţa sistemelor inteligente. - Absolventul este capabil să identifice probleme complexe şi să examineze probleme conexe pentru a dezvolta opţiuni de rezolvare şi implementa soluţii.
Responsabilităţi şi autonomie	- Absolventul are capacitatea de a alege şi folosi paradigme de programare (procedural, orientat obiect, funcţional) pentru realizarea de aplicaţii software adecvate specificului domeniului aplicaţiei dezvoltate. - Absolventul are aptitudinile necesare pentru a aplica diferite metode şi instrumente de analiza şi vizualizare a rezultatelor algoritmilor si tehnicilor de inteligenţă artificială utilizate.

7. Obiectivele disciplinei (reieşind din grila competenţelor acumulate)

7.1 Obiectivul general al disciplinei	• Cunoasterea, intelegerea si utilizarea conceptelor teoretice folosite in proiectarea compilatoarelor • Abilitati avansate de programare
7.2 Obiectivele specifice	• Acumularea de cunostinte despre compilatoare • Intelegerea functionarii unui compilator, depanarea programelor, raportarea erorilor • Intelegerea conceptelor de limbaje formale si dezvoltarea abilitatilor modelarea problemelor folosind limbaje formale; abilitatea de a aplica tehnici specifice compilarii in rezolvarae problemelor din viata reala

8. Conţinuturi

8.1 Curs	Metode de predare	Observaţii

1. Structura generala a unui compilator. Introducere	Expunere, descriere, explicatie, exemple	
2. Analiza lexicala. Limbaje formale	Expunere, descriere, explicatie, exemple	
3. Gramatici. Ierarhia Chomsky. Automate finite	Expunere, descriere, explicatie, exemple	
4. Limbaje regulare. Generator de analizor lexical	Expunere, descriere, explicatie, exemple	
5. Proprietati de inchidere pentru limbaje regulare	Expunere, descriere, explicatie, exemple	
6. Gramatici independente de context	Expunere, descriere, explicatie, exemple	
7. Generator de analiza sintactica. Automate push-down	Expunere, descriere, explicatie, exemple	
8. Gramatici de atribute	Expunere, descriere, explicatie, exemple	
9 si 10. Analiza sintactica	Expunere, descriere, explicatie, exemple	
11 si 12. Generare de cod intermediar si cod obiect	Expunere, descriere, explicatie, exemple	
13 & 14 Sumarizarea aspectelor teoretice si practice. Aplicatii ale proiectarii compilatoarelor	Expunere, descriere, explicatie, exemple	

Bibliografie

1. A.V. AHO, D.J. ULLMAN - Principles of computer design, Addison-Wesley, 1978.
2. A.V. AHO, D.J. ULLMAN - The theory of parsing, translation and compiling, Prentice-Hall, Engl. Cliffs., N.J., 1972, 1973.
3. D. GRIES - Compiler construction for digital computers,, John Wiley, New York, 1971.
4. MOTOGNA, S. – Metode de proiectare a compilatoarelor, Ed. Albastra, 2006
5. SIPSER, M., Introduction to the theory of computation, PWS Pulb. Co., 1997
6. CSÖRNYEI ZOLTÁN, Bevezetés a fordítóprogramok elméletébe, I, II., ELTE, Budapest, 1996
7. L.D. SERBANATI - Limbaje de programare si compilatoare, Ed. Academiei RSR, 1987.

8.2 Seminar	Metode de predare	Observaţii
1. Specificarea unui limbaj de programare; notatia BNF	Dialog, dezbatere, studii de caz, exemple, demonstratii	
2. Automate finite	Dialog, dezbatere, studii de caz, exemple, demonstratii	
3. Gramatici regulare si independente de context	Dialog, dezbatere, studii de caz, exemple, demonstratii	
4 si 5 Proprietati ale limbajelor regulare	Dialog, dezbatere, studii de caz, exemple, demonstratii	
6. Analizor sintactic LR(0)	Dialog, dezbatere, studii de caz, exemple, demonstratii	
7. Analizor sintactic SLR	Dialog, dezbatere, studii de caz, exemple, demonstratii	
8. Analizor sintactic LR(10 si LALR	Dialog, dezbatere, studii de caz, exemple, demonstratii	
9. Automat push dowm	Dialog, dezbatere, studii de caz, exemple, demonstratii	
10. Analizor sintactic LL(1)	Dialog, dezbatere, studii de caz, exemple, demonstratii	
11. Gramatici de atribute	Dialog, dezbatere, studii de caz, exemple, demonstratii	
12. Cod intermediar	Dialog, dezbatere, studii de caz, exemple, demonstratii	
13. Proprietati ale limbajelor independente de context	Dialog, dezbatere, studii de caz, exemple, demonstratii	
14. Exercitii de sumarizare	Dialog, dezbatere, studii de caz, exemple, demonstratii	

8.3 Laborator	Metode de predare	Observaţii
1. Task 1: Specificarea unui mini-limbaj şi implementare analiză lexicală 1.1: Specificare mini limbaj (notaţie BNF)	Explicaţie, dialog, studii de caz	
2. Task 1: Specificarea unui mini-limbaj şi implementare analiză lexicală 1.2: Implementare functii principale	Explicaţie, dialog, studii de caz	
3. Task 1: Specificarea unui mini-limbaj şi implementare analiză lexicală 1.3: Implementare tabela de simboluri	Explicaţie, dialog, studii de caz	
4. Task 1: Specificarea unui mini-limbaj şi implementare analiză lexicală 1.4: Program principal, testare, predare	Explicaţie, dialog, studii de caz	
5. Task 2: Gramatici regulare + AF + transformări 2.1: Definire structuri de date pentru GR şi AF; implementare transformări	Explicaţie, dialog, studii de caz	
6. Task 2: Gramatici regulare + AF + transformări 2.2: Program principal, testare, predare	Discuţie date de testare, evaluare	
7. Task 3: GIC + transformări echivalente 3.1: Extindere task 2 pentru GIC; implementare transformări	Explicaţie, dialog, studii de caz	
8. Task 3: GIC + transformări echivalente 3.2: Program principal, testare, predare	Discuţie date de testare, evaluare	
9. Task 4: Implementare analiză sintactică 4.1: Definire structuri de date şi arhitectura	Explicaţie, dialog, studii de caz	Unul dintre: descendent cu reveniri, LL(1), LR(0), SLR
10. Task 4: Implementare analiză sintactică 4.2: implementare funcţii analiză sintactică	Explicaţie, dialog, studii de caz	Task 4 în echipă de 2 studenţi
11. Task 4: Implementare analiză sintactică 4.3: Program principal şi integrare module	Explicaţie, dialog, studii de caz	
12. Task 4: Implementare analiză sintactică 4.4: testare pe gramatică	Discuţie date de testare, evaluare	
13. Task 4: Implementare analiză sintactică 4.5: testare pe mini limbaj şi secvenţă; predare	Discuţie date de testare, evaluare	
14. Task 5: Instrumente lex şi yacc: 5.1: implementare şi predare	Discuţie date de testare, evaluare	

Bibliografie

1. A.V. AHO, D.J. ULLMAN - Principles of computer design, Addison-Wesley, 1978.
2. A.V. AHO, D.J. ULLMAN - The theory of parsing, translation and compiling, Prentice-Hall, Engl. Cliffs., N.J., 1972, 1973.
3. MOTOGNA, S. – Metode de proiectare a compilatoarelor, Ed. Albastra, 2006
4. G. MOLDOVAN, V. CIOBAN, M. LUPEA - Limbaje formale si automate. Culegere de probleme, Univ. Babes-Bolyai, Cluj-Napoca, 1996

Explica

9. Coroborarea conţinuturilor disciplinei cu aşteptările reprezentanţilor comunităţii epistemice, asociaţiilor profesionale şi angajatori reprezentativi din domeniul aferent programului

- Cursul respectă recomandările de curriculă ACM şi IEEE pentru studii de licenta;
- Cursul există la majoritatea programelor de studii similare la universităţi de prestigiu din România şi străinătate;
- Conţinutul cursului este considerat important de către companii pentru acumularea de cunoştinăe de programare

10. Evaluare

Tip activitate	10.1 Criterii de evaluare	10.2 metode de evaluare	10.3 Pondere din nota finală
10.4 Curs	- cunoaşterea principiilor de bază din domeniu - Aplicarea conceptelor de la curs în rezolvarea de probleme	Examen scris	60%

10.5 Seminar/laborator	- abilitatea de a aplica algoritmi şi a înţelege exemple – rezolvare de probleme	Probleme rezolvate; teme; observare continuă pe parcursul semestrului	10%
	-Abilitatea de implementa conceptele şi algoritmii cursului - aplicarea tehnicilor pentru diferite clase de limbaje de programare	Examinare practică continuă pe parcursul semestrului – documentaţie, portofoliu Github	30%
10.6 Standard minim de performanţă			
• Participarea la minim 75% din activitatea de seminar şi minim 90% din activitatea de laborator pe timpul semestrului • Minim nota 5 (pe o scară de la 1 la 10) la examenul scris şi activitatea de laborator • Înţelegerea conceptelor de bază din limbaje formale: gramatică, automat finit, automat push down, expresii regulare; înţelegerea principiilor compilării, analiză lexicală şi sintactică			

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

Nu se aplică.

Data completării:	Semnătura titularului de curs	Semnătura titularului de seminar
12.04.2025	Prof.PhD. Simona Motogna	Prof.PhD. Simona Motogna

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.*".