FIŞA DISCIPLINEI

Metode avansate de programare funcţională

Anul universitar 2025-2026

1. Date despre program

1.1. Instituţia de învăţământ superior	Universitatea Babeş-Bolyai din Cluj-Napoca
1.2. Facultatea	Facultatea de Matematică şi Informatică
1.3. Departamentul	Departamentul de Matematică şi Informatică al Liniei Maghiare
1.4. Domeniul de studii	Calculatoare şi tehnologia informaţiei
1.5. Ciclul de studii	Licenţă
1.6. Programul de studii / Calificarea	Ingineria informaţiei (în limba maghiară)
1.7. Forma de învăţământ	Învăţământ cu frecvenţă

2. Date despre disciplină

2.1. Denumirea disciplinei	**Metode avansate de programare funcţională / Haladó funkcionális programozás / Advanced methods of functional programming**					Codul disciplinei	**MLM5047**
2.2. Titularul activităţilor de curs	Horváth Zoltán						
2.3. Titularul activităţilor de seminar	Şuteu-Szöllősi Ştefan Lucian						
2.4. Anul de studiu	3	2.5. Semestrul	6	2.6. Tipul de evaluare	Examen (E)	2.7. Regimul disciplinei	Disciplină de specialitate (DS)

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

3.1. Număr de ore pe săptămână	**4**	din care: 3.2. curs	**2**	3.3. seminar/ laborator/proiect	**2**
3.4. Total ore din planul de învăţământ	**56**	din care: 3.5. curs	**28**	3.6 seminar/laborator/proiect	**28**
Distribuţia fondului de timp pentru studiul individual (SI) şi activităţi de autoinstruire (AI)					**ore**
Studiul după manual, suport de curs, bibliografie şi notiţe (AI)					26
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate şi pe teren					15
Pregătire seminare/ laboratoare/ proiecte, teme, referate, portofolii şi eseuri					14
Tutorat (consiliere profesională)					10
Examinări					4
Alte activităţi					
3.7. Total ore studiu individual (SI) şi activităţi de autoinstruire (AI)				**69**	
3.8. Total ore pe semestru				**125**	
3.9. Numărul de credite				**5**	

4. Precondiţii (acolo unde este cazul)

4.1. de curriculum	Nu e cazul
4.2. de competenţe	Cunoştinţe de bază de programare, cunoştinţe de bază de logică matematică, cunoştinţe elementare de programare funcţională

5. Condiţii (acolo unde este cazul)

5.1. de desfăşurare a cursului	Sală de curs cu tablă şi videoproiector
5.2. de desfăşurare a seminarului/ laboratorului	Sală de laborator cu tablă şi videoproiector

6.1. Competenţele specifice acumulate[1]

Competenţe profesionale/esenţiale	• C1.1 Descrierea adecvată a paradigmelor de programare şi a mecanismelor de limbaj specifice, precum şi identificarea diferenţei dintre aspectele de ordin semantic şi sintactic. • C1.5 Dezvoltarea de unităţi de program şi elaborarea documentaţiilor aferente • C4.1 Definirea conceptelor şi principiilor de bază ale informaticii, precum şi a teoriilor şi modelelor matematice • C4.3 Identificarea modelelor şi metodelor adecvate pentru rezolvarea unor probleme reale
Competenţe transversale	• CT1 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 • CT3 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

Cunoştinţe	• Studentul cunoaşte limbajul de programare Haskell şi/sau Clean, noţiunea de curry-ing şi deducţia tipurilor.
Aptitudini	• Studentul utilizează paradigmele programării declarative/funcţionale. • Studentul recunoaşte elementele programării declarative (de exemplu construirea listelor cu „list comprehension”).
Responsabilităţi şi autonomie	• Studentul este capabil să lucreze independent într-unul dintre limbajele funcţionale.

7. Obiectivele disciplinei (reieşind din grila competenţelor acumulate)

7.1 Obiectivul general al disciplinei	• Limbajele funcţionale necesită un model de programare diferit de cel imperativ, iar scopul este de a învăţa şi de a stăpâni aceste strategii de programare. • Aplicarea modelului de programare funcţională în algoritmii moderni • Aprofundarea gândirii despre tipuri şi conştientizarea utilităţii tipurilor
7.2 Obiectivele specifice	• Modelul de programare funcţională; algoritmi recursivi, rescrierea automată a algoritmilor recursivi, derivarea tipurilor. • Definiţii ale sistemelor de tipuri în Haskell/Clean, mecanismul de derivare a tipurilor

[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.

8. Conţinuturi

8.1 Curs	**Metode de predare**	**Obs.**
1. Introducere în paradigma funcţională, diferite limbaje de programare funcţională, comparaţii, caracteristici funcţionale în alte limbaje de programare.	Prelegere, explicaţii, exemplificare	
2. Elemente ale limbajului Haskell/Clean, conceptul de tipuri, mediul de programare, recursivitate.	Prelegere, explicaţii, exemplificare	
3. Manipularea listelor; reprezentarea listelor, construirea lor, operaţii cu liste;	Prelegere, explicaţii, exemplificare	
4. Funcţii de ordin superior, funcţiile map, filter, until, etc.	Prelegere, explicaţii, exemplificare	
5. Operatori, tipuri de funcţii, derivarea tipurilor, funcţii lambda.	Prelegere, explicaţii, exemplificare	
6. Modelul calculului lambda pentru evaluarea unei funcţii/program. Exemple de lambda-calculus, perspectivă teoretică.	Prelegere, explicaţii, exemplificare	
7. Programe de înaltă performanţă, reducerea complexităţii cu zip, zipwith	Prelegere, explicaţii, exemplificare	
8. Funcţiile foldr şi foldl, paradigma map+fold = map-reduce. Modelul de programare pipeline şi aplicaţiile sale.	Prelegere, explicaţii, exemplificare	
9. Tipuri existenţiale, tipuri de ordin superior	Prelegere, explicaţii, exemplificare	
10. Clase de construcţie ale tipurilor	Prelegere, explicaţii, exemplificare	
11. Programare generativă	Explicaţii	
12. Recapitulare, Discuţii despre exerciţiile de verificare.	Explicaţii, exemplificare	
13. Exerciţii interesante, perspective asupra altor limbaje de programare. Scala ca limbaj de programare funcţional	Prelegere, exemplificare	
14. Perspective asupra altor limbaje de programare: limbajul de programare F#.	Prelegere, exemplificare	

Bibliografie

[1]. Bird R (2011) Pearls of Functional Algorithm Design,
[2]. Bird R. (2015) Thinking Functionally with Haskell, Cambridge University Press
[3]. Reede, C. (1989) Elements of Functional Programming, Addison Wesley.
[4]. Petricek T, Skeet J (2009) Real-World Functional Programming, With examples in F# and C#, Manning Publications.
[5]. Field A. (1988) Functional Programming, Addison Wesley, New York.
[6]. Horváth Zoltán (ELTE programnyelvek tanszék) Funkcionális programozás előadása.
[7]. Graham Hutton (2007) Programming in Haskell, Cambridge University Press.
[8]. Miran Lipovaca (2011) Learn you a Haskell for Great Good, No Starch Press, San Francisco.

8.2 Seminar / laborator	**Metode de predare**	**Obs.**
1. Elemente ale limbajului Haskell/Clean, mediul de programare	Explicaţii, exemple	
2. Operaţii aritmetice, liste, constructori de liste	Explicaţii, exemple, rezolvarea problemelor, implementare	
3. Exersarea stilului funcţional, utilizând funcţii precum map, fold, filter etc.	Explicaţii, exemple, rezolvarea problemelor, implementare	
4. Operatori, variabile de tip, polimorfism	Explicaţii, exemple, rezolvarea problemelor, implementare	

5. Tipurile record, tablou, noţiunea de „uniqueness type” (Clean)	Explicaţii, exemple, rezolvarea problemelor, implementare	
6. Tipuri de date algebrice, tratarea erorilor	Explicaţii, exemple, rezolvarea problemelor, implementare	
7. Exerciţii recapitulative	Rezolvarea problemelor, implementare	

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

- Cursul urmează structura cursurilor de la universităţi de top din Anglia (University College London, University of St Andrews) şi SUA (MIT, Stanford).
- Exerciţiile au fost elaborate folosind materialele universităţilor menţionate mai sus – Stanford, MIT, UCL.

10. Evaluare

Tip activitate	10.1 Criterii de evaluare	10.2 Metode de evaluare	10.3 Pondere din nota finală
10.4 Curs	Cunoaşterea conceptelor de bază ale paradigmei programării funcţionale	Examen scris (suplimentat cu întrebări orale, dacă este necesar)	50%
10.5 Seminar/laborator	Abilităţi de programare în Haskell/Clean	Prezentarea temelor (25%) + examen de laborator (25%)	50%
10.6 Standard minim de performanţă			

- Nota de trecere este 5
- Cerinţele minime trebuie îndeplinite atât la examenul scris, cât şi la examenul practic (prezentarea temelor şi examenul de laborator, fiecare în parte) pentru a promova.

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

Data completării:
19.09.2025

Semnătura titularului de curs

Prof. dr. Horváth Zoltán

Semnătura titularului de seminar

Conf. dr. Şuteu-Szöllősi Ştefan Lucian

Data avizării în departament:

Semnătura directorului de departament

Conf. dr. András Szilárd Károly

[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.*".