FIŞA DISCIPLINEI

PROGRAMARE LOGICĂ ŞI FUNCŢIONALĂ

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 Matematică şi Informatică a Liniei Maghiare**
1.4 Domeniul de studii	**Informatică**
1.5 Ciclul de studii	**Licenţă**
1.6 Programul de studiu / Calificarea	**Informatică (în limba maghiară)**

2. Date despre disciplină

2.1 Denumirea disciplinei (ro), (hu), (en)	**Programare logică şi funcţională (ro) /** Logikai és funkcionális programozás (hu) / Logic and functional programming (en)						
2.2 Titularul activităţilor de curs				**Prof. dr. Lehel CSATÓ**			
2.3 titularul activităţilor de seminar				**Prof. dr. Lehel CSATÓ**			
2.4 Anul de studiu	**2**	2.5 Semestrul	**3**	2.6. Tipul de evaluare	**Colocviu**	2.7 Regimul disciplinei	**Obligatoriu - specialitate**
2.8 Codul disciplinei				**MLM5009**			

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	**2**
3.4 Total ore din planul de învăţământ	**56**	Din care: 3.5 curs	**28**	3.6 seminar / laborator	**28**

Distribuţia fondului de timp:	Ore
Studiul după manual, suport de curs, bibliografie şi notiţe	**26**
Documentare suplimentară în bibliotecă şi pe platformele electronice de specialitate	**10**
Pregătire seminare/laboratoare, teme, referate, portofolii şi eseuri	**16**
Tutorat	**14**
Examinări	**3**
Alte activităţi:	**-**

3.7 Total ore studiu individual	**69**
3.8 Total ore pe semestru	**125**
3.9 Numărul de credite	**5**

4. Precondiţii (acolo unde este cazul)

4.1 de curriculum	Elemente de programare
4.2 de competenţe	Abilităţi de înţelegere a conceptelor de bază

5. Condiţii (acolo unde este cazul)

5.1 De desfăşurare a cursului	• Proiector. • Tablă.
5.2 De desfăşurare a Seminarului / laboratorului	• Calculatoare, tablă;

6.1. Competenţele specifice acumulate

Competenţe profesionale	• 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. • C1.3 Elaborarea codurilor sursă adecvate şi testarea unitară a unor componente într-un limbaj de programare cunoscut, pe baza unor specificaţii de proiectare date, • C4.1 Definirea conceptelor şi principiilor de bază ale informaticii, precum şi a teoriilor şi modelelor matematice, • C4.2 Interpretarea de modele matematice şi informatice (formale)
Competenţe transversale	• CT1 Aplicarea regulilor de muncă organizată şi eficientă, a unor atitudini responsabile fată 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 este capabil să citească şi să înţeleagă coduri sursă scrise în Prolog, să descifreze funcţionarea acestora. • Studentul cunoaşte sistemul Haskell, noţiunea de curry-ing, the parametrizare parţială a funcţiilor, de sistemele de deducţie a tipurilor.
Aptitudini	• Studentul recunoaşte şi aplică paradigmele programării declarative, de ex. ”list comprehension”, auto-types. • Studentul ştie să aplice paradigmele programării funcţionale.

Responsabilităţi şi autonomie	Studentul are capacitatea de a lucra independent în sistemele Prolog, Haskell, să înţeleagă mai bine caracteristicile declarative a anumitor sisteme.

7. Obiectivele disciplinei (reieşind din grila competenţelor acumulate)

7.1 **Obiectivul general al disciplinei**	• Obiectivul cursului este familiarizarea cu paradigmele de programare declarativă. • Înţelegerea modelelor de program care sunt capabile să "compileze" program bazate pe specificaţii. • Însuşirea metodologilor şi schemelor cognitive necesare dezvoltării unui sistem informatic în stil declarativ.
7.2 **Obiective specifice**	• Folosirea de noi limbaje de programare: o Limbajul PROLOG pentru programare logică, o Limbajul Haskell pentru programare funcţională; • Definirea aplicabilităţii conceptelor însuşite • Analiza diferitelor concepte de programare, rezolvarea problemelor în stil funcţional, • Specificarea formală a programelor cu ajutorul calculului lambda. • Folosirea conceptului de deducţie automată a tipului.

8. Conţinuturi

8.1 Curs	**Metode de predare**	**Observaţii**
Săptămâna 1. Prezentarea paradigmelor de programare imperativă vs declarativă. Introducere în modelele de programare declarativă.	Expunerea, Conversaţia, Algoritmizarea, Problematizarea	
S.2. Bazele programării logice: definirea regulilor şi a faptelor în limbajul Prolog. Noţiunea de variabile libere şi de variabile folosite (bound); principiul rezoluţiei, a unificării clauzelor. Sistemul de deducţie în Prolog.		
S.3. Evaluarea expresiilor cu variabile în Prolog. Predicate compuse, algoritmul backtracking. Liste în Prolog.		
S.4. Reguli de "pattern matching" în Prolog. Noţiunea de variabilă atomică. Operaţiile aritmetice şi operaţiile logice.		
S.5. Negaţia în Proog. Reguli de descompunere a predicatelor. Reguli de unificare a acestora. Evaluarea predicatelor Prolog.		
S.6. Structuri de date recurente în Prolog. Recapitularea noţiunilor importante a paradigmei de programare logică.		

S.7. Programarea funcţională: definiţii ale conceptelor de bază, prezentarea generală a caracteristicilor limbajelor de programare funcţională.		
S.8. Elementele limbajului de programare funcţională Haskell. Codurile recurente. Noţiunea de tip în Haskell.		
S.9. Liste în Haskell: reprezentarea internă şi folosirea reprezentărilor în codul sursă. Constructori şi generatori de listă.		
S.10. Operatori în Haskell. Definiţia tipurilor funcţiilor. Sisteme de deducţie a tipului unei expresii. Funcţii lambda în Haskell. Definiţia de tipuri noi în Haskell.		
S.11. Introducere în modelele matematice a programelor şi în calculul lambda. Exemple din calculul lambda.		
S.12. Exemple Haskell pentru aplicaţii ale calculului lambda.		
S.13. Recapitularea paradigmelor de programare logică şi funcţională prin exemple. Rezolvare de probleme cu ajutorul limbajului Prolog respectiv al limbajului Haskell.		
S.14. Colocviu,	Probă scrisă	

Bibliografie

[1]. Serban G., Pop H.F. (2006) Elemente avansate de programare in Lisp si Prolog. Aplicatii in Inteligenta Artificiala, Editura Albastra.

[2]. Ásványi Tibor - ELTE - logikai programozás oldalai: Prolog (http://aszt.inf.elte.hu/~asvanyi/pl/jegyzetek, látogatva 2015. május 4-én).

[3]. Prolog könyv - letölthető Mike Spivey oldaláról, (http://spivey.oriel.ox.ac.uk/mike/logic/index.html)

[4]. (***) Learn Prolog Now! (http://www.coli.uni-saarland.de/~kris/learn-prolog-now/)

[5]. Szeredi Péter és Benkő Tamás „Nagyhatékonyságú Logikai Programozás”

[6]. Szeredi P, Benkő T (2001) Deklaratív programozás, Számítástudományi és Információelméleti Tanszék (Budapest), Tűzött kötés , 226 oldal.

[7]. Allen C, Moronuki J (2016) Haskell programming from first principles (draft).

[8]. Reede, C. (1989) Elements of Functional Programming, Addison Wesley.

[9]. Field A. (1988) Functional Programming, Addison Wesley, New York.

[10]. Horváth Zoltán (ELTE programnyelvek tanszék) Funkcionális programozás előadása.

[11]. Graham Hutton (2007) Programming in Haskell, Cambridge University Press.

[12]. Miran Lipovaca (2011) Learn you a Haskell for Great Good, No Starch Press, San Francisco.

8.2 Seminare / Laboratoare	**Metode de predare**	**Observaţii**
Seminare:		
Săptămâna 1. Exemplu re reguli în Prolog: baze de date genealogice,	ia, Algoritmizarea, Pr	

interogările bazelor de reguli. Folosirea listelor.		
S.2. Folosirea de PATTERN-uri în Prolog. Dezvoltare de programe / interogări eficiente.		
S.3. Folosirea predicatelor FINDALL, SETOF, BAGOF. Implementarea algoritmilor de căutare şi sortare în Prolog.		
S.4. Comparaţii între paradigma logică şi cea funcţională. Evidenţierea diferenţelor, exemple de ”transcriere” al unui cod Prolog în Haskell.		
S.5. Exerciţii cu funcţiile de nivel înalt: FILTER, MAP, FOLDR, UNTIL.		
S.6. Deducţia tipurilor unei funcţii, exemple Haskell. Declaraţia de tip. Declaraţia instanţelor.		
S.7. Concepte avansate: structura de monad şi structuri algebrice. (sub formă de prezentări de către studenţi)		
Laboratoare		
Săptămâna 1. Sistemul Prolog. Enunţul şi discuţia primului set de exerciţii.	Investigaţia, Observarea sistematică a studentului	
S2. Verificare soluţii, enunţul celui de-al doilea set de exerciţii.		
S3. Verificare soluţii, enunţ şi discuţii probleme obligatorii şi opţionale.		
S4. Verificare soluţii, enunţ şi discuţii probleme obligatorii şi opţionale.		
S5. Verificare soluţii, enunţ şi discuţii probleme obligatorii şi opţionale.		
S6. Verificare soluţii, enunţ şi discuţii probleme obligatorii şi opţionale.		
S7. Verificare, sumar a punctelor tari şi a celor slabe.		

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

- Predarea sistemului Prolog – ca introducere în programarea declarativă – este metoda generală în universităţile de prestigiu (ex. Univ din Amsterdam, Universitatea ELTE Budapesta).
- Haskell este limbajul de programare ales pentru programarea funcţională atât în predare cât şi în cercetare (ex. Universitatea St.Andrews din Scoţia, Universitatea ELTE Budapesta, Univ. Chalmers din Goteburg).

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 a programării logice, - Calculul rezultatului unei expresii - Cunoaşterea conceptelor programării funcţionale, - evaluarea recurenţei, - evaluarea tipului unei expresii.	Examen scris	60% +10% (claritate)
10.5 Seminar / Laborator	- prezentarea problemelor se laborator. - activitate seminare (rez. probleme la tablă)	Verificare program şi evaluarea calităţii sistemului.	40%
	Rezolvarea problemelor opţionale.	Verificare program şi evaluarea calităţii soluţiilor.	+10%
10.6 Standard minim de performanţă			
• Calculul rezultatelor unui cod sursă într-un limbaj declarativ. • Implementarea în limbajele instruite al unui algoritm simplu.			

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

Nu se aplică	

Data completării | Semnătura titularului de curs | Semnătura titularului de seminar

10.04.2025. | Prof. dr. Lehel CSATÓ | Prof. dr. Lehel CSATÓ

Digitally signed by LEHEL CSATO
Date: 2025.09.17 23:13:07 +03'00'

Data avizării în departament | Semnătura directorului de departament

24.04.2025 | Conf. dr. András Szilárd