

FIȘA DISCIPLINEI

1. Date despre program

1.1 Instituția de învățământ superior	Universitatea Tehnică din Cluj-Napoca
1.2 Facultatea	Automatică și Calculatoare
1.3 Departamentul	Calculatoare
1.4 Domeniul de studii	Calculatoare și Tehnologia Informației
1.5 Ciclul de studii	Licență
1.6 Programul de studii / Calificarea	Calculatoare română / Inginer
1.7 Forma de învățământ	IF – învățământ cu frecvență

2. Date despre disciplină

2.1 Denumirea disciplinei	Programare funcțională				Codul disciplinei	32
2.2 Titularul de curs	Conf.dr.ing. Slăvescu Radu-Răzvan - Radu.Razvan.Slavescu@cs.utcluj.ro					
2.3 Titularul activităților de seminar / laborator / proiect	Prof. dr. ing. Pinteș Camelia - Camelia.Pinteș@mi.utcluj.ro As. drd. ing. Szomiu Roxana - Roxana.Szomiu@cs.utcluj.ro					
2.4 Anul de studiu	III	2.5 Semestrul	5	2.6 Tipul de evaluare	E	
2.7 Regimul disciplinei	Categorii formative: <i>DF – fundamentală, DS - de specialitate, DC - complementară</i>					DF
	Opționalitate: <i>DOB - obligatorie, DOP - opțională, DFA - facultativă</i>					DOB

3. Timpul total estimat

3.1 Număr de ore pe săptămână	4	din care:	Curs	2	Seminar	-	Laborator	2	Proiect	-
3.2 Număr de ore pe semestru	56	din care:	Curs	28	Seminar	-	Laborator	28	Proiect	-
3.3 Distribuția fondului de timp (ore pe semestru) pentru:										
(a) Studiul după manual, suport de curs, bibliografie și notițe										18
(b) Documentare suplimentară în bibliotecă, pe platforme electronice de specialitate și pe teren										10
(c) Pregătire seminarii / laboratoare, teme, referate, portofolii și eseuri										10
(d) Tutoriat										4
(e) Examinări										2
(f) Alte activități:										-
3.4 Total ore studiu individual (suma (3.3(a)...3.3(f)))				44						
3.5 Total ore pe semestru (3.2+3.4)				100						
3.6 Numărul de credite				4						

4. Precondiții (acolo unde este cazul)

4.1 de curriculum	Cursul de Structuri de Date și Algoritmi
4.2 de competențe	Nu sunt necesare cunoștințe anterioare de Programare Funcțională, dar este recomandat să existe o experiență de minimum un an de programare într-un limbaj cum ar fi Java, C sau C++

5. Condiții (acolo unde este cazul)

5.1. de desfășurare a cursului	Tabla, proiector, calculator
5.2. de desfășurare a laboratorului	Calculatoare, interpretoare/compilatoare pentru limbajele studiate Prezență obligatorie 100% pentru admiterea la examenul final

6. Competențele specifice acumulate

6.1 Competențe profesionale	C2 - Proiectarea unui sistem software în maniera funcțională • C2.1 – Identificarea și descrierea componentelor software ale sistemului • C2.2 - Explicarea rolului fiecăre componente și a interacțiunii acestora. • C2.3 - Construirea unor componente software folosind metode de proiectare, limbaje, tehnologii și instrumente specifice Programării Funcționale. • C2.4 - Implementarea componentelor software pe baza de cod în stil funcțional, idiomatic și eficient. • C2.5 - Evaluarea caracteristicilor funcționale și nefuncționale ale sistemului software pe baza unor metrice de performanță și demonstrarea corectitudinii sale.
6.2 Competențe transversale	N/A

7. Rezultatele așteptate ale învățării

Cunoștințe	Describe, identifică, sumarizează concepte și metode elementare privitoare la limbaje de programare, medii de programare, tehnici de programare, baze de date, inteligență artificială și inginerie software și modul lor de aplicare în probleme concrete.
Aptitudini	Alege și explică concepte proprii specifice proiectării algoritmilor, programării orientate pe obiecte, programării logice și funcționale. Specifică cerințe, analizează, elaborează, dezvoltă și testează programe în limbaje de programare de uz general (C, etc.) și /sau obiect-orientate (C++, Java, etc.), aplicând elementele specifice ingineriei software. Proiectează, implementează, dezvoltă și /sau gestionează aplicații ce includ diverse tipuri de baze de date.
Responsabilitate și autonomie:	Are o comportare onorabilă, responsabilă, etică, în spiritul legii pentru a asigura reputația profesiei.

8. Obiectivele disciplinei

8.1 Obiectivul general al disciplinei	Dezvoltarea abilității de scrie cod corect, concis, pe baza elementelor paradigmei funcționale (imutabilitatea variabilelor, nivel ridicat de abstractizare, posibilitatea demonstrării formale a corectitudinii unui program, posibilitatea paralelizării facile a codului), precum și a fundamentelor sale teoretice (calculul lambda)
8.2 Obiectivele specifice	Pentru atingerea acestor obiective generale, studenții vor învăța: - sa scrie cod în maniera funcțională, cu eliminarea variabilelor de stare - sa identifice avantajele și dezavantajele diferitelor stiluri de programare - sa utilizeze recursivitatea și sa structureze codul pentru optimizarea sa - sa folosească funcțiile de ordin superior - sa exploateze mecanismele evaluării lenese și structurile infinite - sa demonstreze formal corectitudinea unui program - sa manipuleze expresii lambda de baza

9. Conținuturi

9.1 Curs	Nr.ore	Metode de predare	Observații
Introducere. Paradigme de programare. Noțiuni fundamentale de programare în Haskell, Elm: funcții, identificatori, recursivitate	2		
Noțiuni fundamentale: recursivitate, constante, tipuri de date primitive, tuple, operatori infix, evaluare	2		
Noțiuni fundamentale: declarații locale, tipuri de date, polimorfism.	2		
Liste: construire listă, operații fundamentale pe liste.	2		

Liste: operatori de liste (generatori, garzi, extensiune listă).	2	Slideuri, Demonstratii si reprezentare modele pe tabla, exercitii rapide pentru cresterea interactiunii	
Arbori: date alternative, potrivire forme, exceptii, arbori binari, conversie arbori-liste.	2		
Arbori: arbori binari de căutare, verificare proprietate arbori echilibrați AVL, printare.	2		
Implementare operații pe mulțimi. Rationator propozitional.	2		
Funcții de ordin superior: funcții anonime, aplicare parțială, relația dintre funcții si date, combinatori	2		
Funcții de ordin superior pentru liste (map, filter, fold)	2		
Date infinite: evaluare leneșă, obiecte infinite, structuri circulare.	2		
Calcul Lambda: notație lambda, conversii, combinatori.	2		
Raționare asupra corectitudinii programelor: inducție structurală, echivalența funcțiilor, inducție pe numărul de noduri.	2		
Monade. Exemple de cazuri de utilizare	2		
Bibliografie (bibliografia minimală a disciplinei conținând cel puțin o lucrare bibliografică de referință a disciplinei, care există la dispoziția studenților într-un număr de exemplare corespunzător)			
1. Haskell - A Purely Functional Language, www.haskell.org 2. Elm – A Delightful language for reliable web applications, elm-lang.org 3. G. Hutton. Programming in Haskell, 2nd edition Cambridge University Press, 2016 4. M. Lipovaca. Learn You a Haskell for Great Good. No Starch Press, 2011. 5. Raul Rojas, A Tutorial Introduction to the Lambda Calculus, FU Berlin, 2015 6. I.A. Letia, L.A. Negrescu, L. Negrescu. Programare functionala, vol. I. Ed. Albastra, Cluj-Napoca, 2006			
9.2 Aplicații - seminar / laborator / proiect	Nr.ore	Metode de predare	Observații
Introducere în Programarea Funcționala folosind Elm	2	Rezolvare de exerciții și probleme, implementare de funcții pe calculator, trasare de algoritmi	
Tipuri în Elm	2		
Liste și recursivitate	2		
Funcții de ordin superior în Elm	2		
Evaluare Elm	2		
Miniaplicatie Elm	2		
Introducere în Haskell. Liste, recursivitate	2		
Verificare de tipuri în Haskell	2		
Arbori în Haskell	2		
Funcții de ordin superior Haskell	2		
Evaluare leneșă și liste infinite	2		
Miniaplicatie în Haskell	2		
Calcul lambda	2		
Evaluare Haskell	2		
Bibliografie (bibliografia minimală pentru aplicații conținând cel puțin o lucrare bibliografică de referință a disciplinei care există la dispoziția studenților într-un număr de exemplare corespunzător)			
1. www.haskell.org 2. elm-lang.org 3. M. Lipovaca. Learn You a Haskell for Great Good. No Starch Press, 2011.			

*Se vor preciza, după caz: tematica seminariilor, lucrările de laborator, tematica și etapele proiectului.

10. Coroborarea conținuturilor disciplinei cu așteptările reprezentanților comunității epistemice, asociațiilor profesionale și angajatorilor reprezentativi din domeniul aferent programului

Conținutul disciplinei este în concordanță cu conținutul unor cursuri similare ale altor universități (Paradigme de Programare, Haskell paralel și concurrent). Cursul evidențiază tehnicile specifice programării funcționale care au pătruns și pătrund în limbajele (multiparadigma) moderne, precum și posibilitatea de a demonstra formal corectitudinea programelor. Studenții sunt încurajați să identifice idei specifice Programării Funcționale prezente în practica curentă a companiilor locale IT.

11. Evaluare

Tip activitate	Criterii de evaluare	Metode de evaluare	Pondere din nota finală
Curs	Înțelegerea conceptelor specifice paradigmei de programare funcțională și bazei sale teoretice. Participare activă la curs, teme.	Examen final scris / Test Moodle	50%
Seminar	-	-	-
Laborator	Cantitatea și calitatea codului Haskell și Elm dezvoltat. Abilitatea de a identifica și corecta erori de programare.	Teste de laborator și mini-aplicații	50%
Proiect	-	-	-
Standard minim de performanță: Înțelegerea și abilitatea de a dezvolta cod în limbaje funcționale, folosind recursivitatea, funcțiile de ordin superior, potrivirea șabloanelor Nota finală: 50% laborator + 50% examen scris Condiții pentru participare la examenul final: Medie Laborator ≥ 5 Condiții pentru promovare: Medie Examen ≥ 5			

Data completării: 24.04.2026	Titulari	Titlu, prenume / NUME	Semnătura
	Curs	Conf.dr.ing. Radu-Răzvan SLĂVESCU	
	Aplicații	Prof.dr.ing. Camelia PINTEA	
		As. drd. ing. Roxana SZOMIU	

Data avizării în Consiliul Departamentului Calculatoare	Director Departament, Prof.dr.ing. Rodica Potolea
Data aprobării în Consiliul Facultății de Automatică și Calculatoare	Decan, Prof.dr.ing. Vlad Mureșan