Universitatea Naţională de Ştiinţă şi Tehnologie Politehnica Bucureşti

Facultatea de Electronică, Telecomunicaţii şi

Tehnologia Informaţiei

FIŞA DISCIPLINEI

1. Date despre program

1.1 Instituţia de învăţământ superior	**Universitatea Naţională de Ştiinţă şi Tehnologie Politehnica Bucureşti**
1.2 Facultatea	**Electronică, Telecomunicaţii şi Tehnologia Informaţiei**
1.3 Departamentul	**Dispozitive, Circuite şi Arhitecturi Electronice**
1.4 Domeniul de studii	Inginerie Electronică, Telecomunicaţii şi Tehnologii Informaţionale
1.5 Ciclul de studii	Licenţă
1.6 Specializarea	Microelectronică, Optoelectronică şi Nanotehnologii

2. Date despre disciplină

2.1 Denumirea disciplinei (ro) (en)				Structuri de date şi algoritmi Data structures and algorithms (Abstract Data Types)			
2.2 Titularul activităţilor de curs				S.l./Lect. Dr. Vlad-Alexandru Grosu			
2.3 Titularul activităţilor de seminar / laborator				S.l./Lect. Dr. Vlad-Alexandru Grosu			
2.4 Anul de studiu	2	2.5 Semestrul	1	2.6. Tipul de evaluare	E	2.7 Regimul disciplinei	Ob
2.8 Tipul disciplinei		D	2.9 Codul disciplinei	04.D.03.O.005		2.10 Tipul de notare	Nota

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

3.1 Număr de ore pe săptămână	3.5	Din care: 3.2 curs	2	3.3 seminar/laborator	1.5
3.4 Total ore din planul de învăţământ	49	Din care: 3.5 curs	28	3.6 seminar/laborator	21
Distribuţia fondului de timp:					ore
Studiul după manual, suport de curs, bibliografie şi notiţe Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate Pregătire seminarii/ laboratoare/proiecte, teme, referate, portofolii şi eseuri					43
Tutorat					4
Examinări					4
Alte activităţi (dacă există):					0

3.7 Total ore studiu individual	51.00
3.8 Total ore pe semestru	100
3.9 Numărul de credite	4

4. Precondiţii (acolo unde este cazul)

4.1 de curriculum	• Programarea Calculatoarelor şi Limbaje de Programare 1 (PCLP1) • Algebră şi Analiză Matematică (necesare înţelegerii proprietăţilor specifice şi modelării colecţiilor dinamice de date studiate: stive, arbori, tabele de dispersie, grafuri)

4.2 de rezultate ale învăţării	Aplicarea cunoştinţelor, conceptelor şi metodelor de bază din algebră şi PCLP1

5. Condiţii necesare pentru desfăşurarea optimă a activităţilor didactice (acolo unde este cazul)

5.1 Curs	Asigurarea de proiector optic (video-proiector) împreună cu toate accesoriile aferente (cabluri de alimentare, date şi semnal video, telecomandă).
5.2 Seminar/ Laborator/Proiect	• Sală dotată cu staţii de lucru de tip calculator personal (desktop/laptop) pe care este instalat suportul software necesar (sistem de operare şi aplicaţii de lucru specific necesar - mediu integrat de lucru). • Prezenţa obligatorie la activităţile de laborator (cerinţă conformă Regulamentelor interne în vigoare privind desfăşurarea activităţilor din POLITEHNICA Bucureşti).

6. Obiectiv general *(Se referă la intenţiile profesorilor pentru studenţi, la ceea ce studenţii vor fi învăţaţi în timpul cursului. Oferă o orientare cu privire la locul cursului în cadrul domeniului ştiinţific abordat, precum şi la rolul pe care acesta îl are în cadrul specializării studiate. Vor fi descrise de o manieră generală tematicile abordate, justificarea includerii cursului în planul de învăţământ al specializării studiate etc.)*

- Curs:

- Înţelegerea jargonului de specialitate folosit în contextul colecţiilor dinamice de date;
- Deprinderea practică a noţiunilor necesare oferite de limbajele de programare, cu accent pe: algoritm, rutină de calcul, alocare dinamică, bibliotecă proprie - împreună cu utilitatea acestora;
- Familiarizarea cu tehnicile de modularizare oferite de limbajul ANSI/ISO C;
- Deprinderea gândirii algoritmice şi exersarea acesteia prin analiza algoritmilor studiaţi şi sinteza de noi algoritmi;
- Posibilitatea de a evalua comparativ diferiţi algoritmii pentru o aceeaşi problemă şi de a-l putea alege pe cel mai bun, pentru a concepe programe optimizate pentru fiecare situaţie din realitate.

- Laborator:

- Dezvoltarea unor rutine generale pentru aplicaţii tipice domeniului programării folosind *colecţii dinamice de date*;
- Utilizarea în conceperea programelor a limbajului ISO C;
- Studiul aspectelor privind portabilitatea programelor;
- Interfaţarea şi lucrul imediat cu majoritatea pachetelor de dezvoltare/proiectare, care se bazează pe tandemul ISO C/C++.

7. Competenţe *(Capacitatea dovedită de a utiliza cunoştinţe, aptitudini şi abilităţi personale, sociale şi/sau metodologice în situaţii de muncă sau de studiu şi pentru dezvoltarea profesională şi personală. Reflectă cerinţele angajatorilor.)*

Specifice	• C3: Aplicarea cunoştinţelor, conceptelor şi metodelor de bază privitoare la arhitectura sistemelor de calcul, microprocesoare, microcontrolere, limbaje şi tehnici de programare. • C4: Utilizarea tehnologiilor şi mediilor de programare

Transversale (generale)	• CT1: Analiza metodică a problemelor întâlnite în activitate, identificând elementele pentru care există soluții consacrate, asigurând astfel îndeplinirea sarcinilor profesionale; • CT3: Adaptarea la noile tehnologii, dezvoltarea profesională și personală, prin formare continuă folosind surse de documentare tipărite, software specializat şi resurse electronice în limba română și, cel puțin, într-o limbă de circulație internațională.

8. Rezultatele învăţării *(Sunt enunţuri sintetice referitoare la ceea ce un student va fi capabil să facă sau să demonstreze la finalizarea unui curs. Rezultatele învățării reflectă realizările studentului și mai puțin intențiile profesorului. Rezultatele învățării informează studenții despre ceea ce se așteaptă de la ei din punct de vedere al performanței, pentru a obține notele și creditele dorite. Sunt definite în termeni concreți, folosind verbe similare exemplelor de mai jos și indică ceea ce se va urmări prin evaluare. Rezultatele învățării vor fi astfel redactate încât să fie evidențiată clar relația față de competențele definite la punctul 7.)*

Cunoştinţe	*Rezultatul asimilării de informații prin învățare. Cunoștințele reprezintă ansamblul de fapte, principii, teorii și practici legate de un anumit domeniu de muncă sau de studiu. Pot fi teoretice și/sau faptice.* • **Enumeră** clasele principale ale de care aparţin colecţiile de date studiate. • **Înţelege** faptul că implementarea nu este singura activitate de bază în contextul modelării pe calculator a problemelor întâlnite. • **Analizează** cerinţele de proiectare şi paşii necesari proiectării unei aplicaţii oarecare. • **Este capabil să descrie** zonele componente ale programului implementat, necesar rezolvării unei probleme date (prezentată în limbaj natural).
Aptitudini	*Capacitatea de a aplica cunoștințe și de a utiliza know-how pentru a duce la îndeplinire sarcini și a rezolva probleme. Aptitudinile sunt descrise ca fiind cognitive (implicând utilizarea gândirii logice, intuitive și creative) sau practice (implicând dexteritate manuală și utilizarea de metode, materiale, unelte și instrumente).* • **Identifică şi descrie** conceptele de bază ale colecţiilor dinamice de date, într-un context dat, împreună cu modalităţile de implementare ale acestora, pentru fiecare clasă de colecţii. • **Modelează** problemele propuse, formalizând textul problemei (exersează etapa de proiectare a programului/aplicaţiei). • **Concepe** noi tipuri de date şi le foloseşte pentru a rezolva probleme legate de colecţiile dinamice de date. • **Exersează** abilităţile de abstractizare şi translatare către limbajul de programare a ideilor exprimate (cuprinse) în enunţul problemei (exprimat în limbaj natural). • **Identifică soluţii** şi **elaborează** planuri de rezolvare pentru problemele propuse (exprimate în limbaj natural). • **Analizează şi compară** soluţia găsită problemei propuse spre rezolvat cu sugestiile oferite de către titularul de laborator pentru problema specifică de rezolvat. • **Testează, depanează şi rulează** programul scris (aplicaţia implementată) şi este capabil să îndrepte posibilele erori, identificându-le în prealabil, alături de posibilele consecinţe (mai ales de ordin semantic).

Responsabilitate și autonomie	*Capacitatea cursantului de a aplica în mod autonom și responsabil cunoștințele și aptitudinile sale.* • **Demonstrează receptivitate** pentru contexte noi de învățare, plecând de la *caracterul integrativ* al disciplinei, bazată pe colecţii dinamice de date. • **Respectă** principiile de etică academică, înțelegând responsabilitățile asumate de rezolvare individuală a problemelor propuse, folosind metodele și tehnicile învățate. • **Demonstrează autonomie** în organizarea situației/contextului de învățare sau a situației problemă de rezolvat. • **Înţelege** implicațiile codului pe care îl scrie, **învaţă** cum să revizuiască codul altor persoane şi apreciază feedback-ul primit. • **Conştientizează** valoarea contribuției sale în domeniul ingineriei software, din perspectiva vieții active, odată cu identificarea și propunerea unor soluții proprii, care să rezolve probleme din viața socială și economică (responsabilitate socială).

9. Metode de predare *(Se vor avea în vedere metode care să asigure predarea centrată pe student. Se va descrie modul în care se asigură participarea studenților la stabilirea propriului parcurs de învățare, cum se identifică eventualele rămâneri în urmă și ce măsuri remediale se adoptă în astfel de cazuri.)*

Interactivitate cu studenții prin intermediul părții aplicative asociate metodelor predate (legătura cu laboratorul).

Sunt rezervate intervale de prezentare și rezolvare a unor probleme practice (întâlnite în viața reală). Partea de modelare se reduce adesea la anunțarea principiilor de rezolvare a unor probleme tipice de programare, care cer rezultate imediate.

Ca suport digital, în prezentare sunt folosite atât platformele recomandate (LMS - Moodle; team management interactiv - MSTeams), precum și alte variante considerate în mod tradițional potrivite pentru susținerea activității didactice (e.g. site-uri suport de programare, site-uri formatoare de conținut, forumuri de specialitate).

Creierul uman are limitări legate de distributivitate: oferă capacitatea de a efectua 4 lucruri deodată. Modularitatea programului și abstractizarea codului sunt obiective esențiale, motiv pentru care toate programele concepute sunt modularizate și folosesc tehnicile specifice C/C++ de izolare a implementării de interfața de programare. Găsirea ierarhiei adecvate (din punct de vedere logic) este un pas fundamental către proiectarea unui sistem complex.

Dat fiind trunchiul comun de programare de care disciplina aparține (în planul de învățământ apare în anul 2), formarea deprinderilor de gândire abstractă și conceptualizare depinde în foarte mare parte de stilul documentației (experiența auditoriului fiind limitată în acest moment). Din această perspectivă, documentația indispensabilă oferită este ușor de utilizat - atât în format fizic, prin sugestiile bibliografice oferite, cât și în format electronic (postate folosind LMS - Moodle).

10. Conținuturi

CURS		
Capitolul	**Conținutul**	**Nr. ore**

1	= Introducere = Algoritmi şi performanţa acestora. Măsurarea timpilor algoritmilor. Calculul valorii unui polinom (varianta clasică vs. schema Horner). Utilitatea practică a colecţiilor dinamice de date.	2
2	= Modularizarea programelor C = Componentele unui program. Modul program. Colecţie de programe - conceptul de Project în scrierea programelor de anvergură.	2
3	= Structuri şi pointeri = Funcţii cu argument structură şi cu tip returnat structură. Tipuri de date generice (utilizator). Modelarea nodului în limbaj C/C++. Liste: generalităţi. Conceptul de înlănţuire, particularităţi. Utilitate.	4
4	= Liste simplu înlănţuite = Liste simplu înlănţuite. Operaţii tipice: parcurgere, inserare element, ştergere element, verificare lista vidă, lungime listă. Sugestii de implementare.	2
5	Liste dublu înlănţuite Utilitate şi concepte specifice. Operaţii tipice: parcurgere flexibilă, inserare nod, ştergere nod, căutare element, test listă vidă, calcul lungime listă. Sugestii de implementare.	2
6	= Liste circulare = Descriere. Particularităţi de construcţie. Reprezentare. Operaţii uzuale: inserare, ştergere, parcurgere, test listă vidă. = Stiva (caz particular de LSI) = Principiu de funcţionare (LIFO). Utilitate, concepte şi termeni specifici. Reprezentare grafică. Operaţii specifice: adăugare element, ştergere element, parcurgere, test stivă vidă, inspecţia elementului din vârful stivei. Sugestii de implementare.	4
7	= Coada (caz particular de LSI) = Principiu de funcţionare (FIFO). Utilitate, concepte şi termeni specifici. Reprezentare grafică. Operaţii specifice: inserare element, ştergere element, inspecţie element, calcul lungime, parcurgere, distrugere coadă, test coadă vidă. Sugestii de implementare.	2
8	= Tabele de dispersie = Principiul de lucru şi concepte specifice. Funcţia de dispersie. Adresare directă. Coliziune. Factor de încărcare. Clasificare (înlănţuite, adresare deschisă). Utilitate. Tipuri de funcţii de dispersie (metoda împărţirii, metoda înmulţirii, funcţii de dispersie dedicate şirurilor). Operaţii specifice: inserare element, căutare element, ştergere element.	3

9	= Arbori (general) şi arbori binari (particularizare) = Utilitate. Termeni şi concepte specifice: nod, gradul arborelui, ramuri, cale şi lungimea acesteia, înălţimea unui nod, înălţime arbore, nivelul unui nod). Reprezentare grafică (convenţii acceptate): apartenenţă, imbricarea parantezelor, forma ierarhică (tradiţional). Modelare algoritmică: referinţe descendente, referinţe ascendente, fiu-frate. = Arbori binari = Definiţie şi caracteristici. Utilitate. Modelare în limbaj. Operaţii tipice: inserare nod, ştergere nod, parcurgere (adâncime, lăţime), căutare informaţie, calcul înălţime. Crearea arborilor binari. Arbore binar echilibrat: mod de construcţie. Parcurgerea arborilor binari (şi echilibraţi): în ordine, preordine, postordine. Parcurgerea pe niveluri (în lăţime). Calcul înălţime pentru arbori binari. Căutarea informaţiei într-un arbore.	4
10	= Introducere în grafuri = Conceptele de bază, tehnici de reprezentare, parcurgere şi variante de implementare. Aplicabilitate în probleme practice.	2
11	Recapitulare: exemple de exerciţii şi probleme.	1
	Total:	28

Bibliografie:

1. Şl. Dr. ing. Vlad-Alexandru GROSU – Structuri de Date şi Algoritmi, suport de curs în format electronic (Moodle) (https://curs.upb.ro/2024/course/view.php?id=3374, https://curs.upb.ro/2023/course/view.php?id=10043)
2. Leon LIVOVSCHI, Horia GEORGESCU – *Sinteza şi analiza algoritmilor*, Ed. Ştiinţifică şi Enciclopedică, Bucureşti, 1986.
3. Florin MUNTEANU, Gh. MUSCĂ, Florin MORARU – *C - Tehnici de programare*, Ed. Joint Printing House, 1995.
4. Emanuela CERCHEZ, Marinel ŞERBAN – *Programare în limbajele C/C++ pentru liceu*, vol. 3, Polirom, 2006.
5. Thomas CORMEN, Charles LEISERSON, Ronald RIVEST – *Introducere în algoritmi*, Computer Libris, Agora, Cluj (2000-2009).
6. Kyle LOUDON – *Mastering algorithms with C*, O'Reilly, 1999.
7. Robert SEDGEWICK, Kevin WAYNE – *Algorithms*, 4th ed., Addison Wesley, 2011.
8. Niklaus WIRTH – *Algorithms + Data structures = Programs*, Prentice Hall, ≥ 1995.
9. Ellis HOROWITZ, Sartaj SAHNI – *Fundamentals of computer algorithms*, Springer, 1983-1998.

LABORATOR

Nr. crt.	Conţinutul	Nr. ore
1	Liste simplu înlănţuite. Conceperea unei biblioteci dedicate.	2
2	Liste dublu înlănţuite. Conceperea unei biblioteci dedicate.	2
3	Testare intermediară şi evaluarea soluţiilor.	2
4	Tabele de dispersie. Soluţie software completă.	2
5	Arbori binari de căutare. Soluţie software completă.	2
6	Testare intermediară şi evaluarea soluţiilor.	2
7	Exemple de probleme rezolvate (liste, stive, cozi)	2

8	Exemple de probleme rezolvate (tabele de dispersie)	2
9	Exemple de probleme rezolvate (arbori binari)	2
10	Verificare finală laborator (colocviu). Identificarea contextului real prezentat în problemă şi adaptarea practică (prin implementare) a conceptelor studiate.	2
11	Test de verificare a abilităţilor de programare.	1
	Total:	21

Bibliografie:

1. Şl. dr. ing. Vlad-Alexandru GROSU – Structuri de Date şi Algoritmi, materiale studiu laborator site personal (https://itlectures.ro/ro_RO/sda/materiale-studiu-sda/) + Moodle: (https://archive.curs.upb.ro/2021/user/files.php#)
2. I. Rusu, Dana Gavrilescu, Vlad Al. Grosu – *Îndrumar de laborator pentru programarea calculatoarelor: C*, Editura MatrixRom, Bucureşti, 2004.
3. Florin Munteanu, Gh. Muscă, Florin Moraru – *C - tehnici de programare*, Editura Joint Printing House, Bucureşti, 1995.
4. Niklaus WIRTH -– *Algorithms + Data structures = Programs*, Prentice Hall, 1995.
5. https://linux.die.net (site suport pentru API-ul C/C++)
6. https://stackoverflow.com/ (forum dedicat discuţiilor practice din zona IT şi conex)

11. Evaluare

Tip activitate	11.1 Criterii de evaluare	11.2 Metode de evaluare	11.3 Pondere din nota finală
11.4 Curs	- Identificarea corectă a contextelor teoretice şi practice de aplicare a algoritmilor predaţi. - Aprofundarea noţiunilor de limbaj necesare cursului, cu aplicare în domeniul ingineriei informaticii şi a tehnologiilor informaţionale (ca domenii de pregătire fundamentală). C4.1: Definirea conceptelor, principiilor şi metodelor folosite în domeniile: programarea calculatoarelor, limbaje de nivel înalt şi specifice, arhitectura sistemelor de calcul, sisteme electronice programabile, grafică, arhitecturi software reconfigurabile.	- Lucrare de verificare programată în pre-sesiune. - Se investighează atât capacitatea studenţilor de a formaliza (identifica noţiunile şi conceptele specifice limbajului studiat, presupuse de problema respectivă) cât şi de a traduce în limbaj (transpunere în program) a unor probleme exprimate în limba română (limbaj natural). - Subiectele acoperă atât partea teoretică aferentă definirii conceptelor colecţiilor dinamice de date, cât şi pe cea practică, din perspectiva capacităţii de rezolvare cu ajutorul limbajului ANSI C/C++ (prin algoritmizare) a unor probleme tipice de programare (prezentate în limba română).	40%

11.5 Seminar/laborator/proiect	C4.5: Susţinerea şi promovarea unei probe referitoare la arhitectura şi principiile practice ale unei structuri software funcţionale. C6.5: Susţinerea unei probe privind stabilirea şi descrierea operaţiilor necesare pentru realizarea şi testarea unui algoritm.	Activitatea de laborator este verificată constant, pe tot parcursul semestrului. Studenţii pot acumula 30%, în urma a două teste de verificare pe parcurs, fiecare a câte de 30 minute. Laboratorul se încheie cu o verificare finală individuală (colocviu), la staţie (pondere: 30%). Aceasta se bazează pe implementarea unui algoritm din materia parcursă pe parcursul întregului semestru şi pe rezolvarea unui subiect teoretic cu rol sintetic.	60%
11.6 Condiţii de promovare			
• Verificarea abilităţilor de *înţelegere şi de identificare* a situaţiilor practice specifice metodelor prezentate precum şi a aplicării corecte a acestora în zona IT/ inginerie şi tehnologii informaţionale. printr-o probă practică de implementare (individuală), prin care se urmăreşte capacitatea de abstractizare şi conceptualizare în termenii unui limbaj de programare şi apoi de alegere a colecţiilor dinamice cele mai potrivite conform specificaţiilor, exprimate în limbaj natural, ale unei probleme reale. • Promovarea materiei presupunea acumularea (fără praguri intermediare impuse) a ***cel puţin 50p*** din totalul de 100p aferent.			

12. Coroborarea conţinutului disciplinei cu aşteptările reprezentanţilor angajatorilor şi asociaţiilor profesionale reprezentative din domeniul aferent programului, precum şi cu stadiul actual al cunoaşterii în domeniul ştiinţific abordat şi practicile în instituţii de învăţământ superior din Spaţiul European al Învăţământului Superior (SEÎS)

În prezent se impune pregătirea viitorilor ingineri şi cercetători în domeniul algoritmilor de rezolvare a problemelor practice de proiectare şi testare software, de administrare a bazelor de date sau chiar a celor care stau la baza nucleului unui sistem de operare. Formarea studenţilor pe trunchiul de programare asigură cunoştinţele fundamentale necesare în orice activitate profesională ulterioară: învăţământ, cercetare şi/sau proiectare.

Activităţile disciplinei *Structuri de Date şi Algoritmi* oferă bazele gândirii algoritmice şi ale tehnicilor specifice aplicării unui limbaj de programare standardizat (se programează în limbajul ISO C/C++) în contextul unor probleme practice din viaţa reală dar şi din contextul sistemelor de calcul (algoritmi întâlniţi în cadrul sistemelor de operare). Plecând de la titlul unei cărţi de referinţă în domeniu, "*Algorithms + Data Structures = Programs*" scrisă de Niklaus Wirth, un program serios nu se poate concepe fără preluarea în implementare a *colecţiilor dinamice de date* (eng. "abstract data types"). Menirea disciplinei este de a fixa cel puţin bazele înţelegerii şi utilizării acestor colecţii de date, antrenând în scrierea programelor şi principii sintactice absolut necesare, deja cunoscute de la disciplinele PCLP1/2, precum administrarea memoriei într-un program - prin intermediul *tehnicilor de alocare dinamică*.

Studenţii deprind utilizarea unui mediu integrat de programare. Variantele utilizate în laborator (DevC++, Code::Blocks) sunt bazate pe portarea MinGW (x86 sau x64) a uneltelor open-source (gcc, g++, gdb, gprof, make) dinspre universul Unix/Linux spre cel closed-source Windows. În variantele oferite, acestea suportă standardul minimal al limbajului C (ISO/IEC 9899:1999, cunoscut drept C99), respectiv standardul minimal C++ (ISO/IEC 14882:2011, denumit C++11).

Prin structurarea informaţiei conform activităţilor prevăzute în curriculum, precum şi prin activitatea de tutoriat desfăşurată, materia oferă paşii necesari aprecierii calităţii, meritelor şi limitelor unor procese, programe, proiecte, concepte, metode şi teorii. Odată cu parcurgerea interesată a disciplinei, studenţii vor avea capacitatea de a discerne (a alege) asupra diverşilor algoritmi necesari în practica de specialitate.

Utilizarea adecvată a criteriilor şi metodelor de evaluare, conforme normelor academice europene la care universitatea POLITEHNICA Bucureşti este parte, permite studenţilor să se autoevalueze continuu (eng. 'personal feedback'), pornind de la calificativele obţinute dar şi ţinând cont de observaţiile şi indicaţiile metodologice pe care titularul de curs/laborator le oferă.

Data completării	Titular de curs	Titular(i) de aplicaţii
25.09.2025	S.l./Lect. Dr. Vlad-Alexandru Grosu 	S.l./Lect. Dr. Vlad-Alexandru Grosu 

Data avizării în departament	Director de departament
26.09.2025	Prof. Dr. Claudius Dan 

Data aprobării în Consiliul Facultăţii	Decan
26.09.2025	Prof. Dr. Mihnea Udrea