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)				Programarea calculatoarelor şi limbaje de programare 2			
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	1	2.5 Semestrul	2	2.6. Tipul de evaluare	E	2.7 Regimul disciplinei	Ob
2.8 Tipul disciplinei		F	2.9 Codul disciplinei	04.F.02.O.011		2.10 Tipul de notare	Nota

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

3.1 Număr de ore pe săptămână	3	Din care: 3.2 curs	2	3.3 seminar/laborator	1
3.4 Total ore din planul de învăţământ	42	Din care: 3.5 curs	28	3.6 seminar/laborator	14
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					52
Tutorat					2
Examinări					4
Alte activităţi (dacă există):					0

3.7 Total ore studiu individual	58.00
3.8 Total ore pe semestru	100
3.9 Numărul de credite	4

4. Precondiţii (acolo unde este cazul)

4.1 de curriculum	Parcurgerea şi/sau promovarea următoarelor discipline: • Programarea Calculatoarelor şi Limbaje de Programare 1 (PCLP1)

4.2 de rezultate ale învățării	Acumularea următoarelor cunoştinţe: • Aplicarea cunoştinţelor, conceptelor şi metodelor de bază privitoare la arhitectura sistemelor de calcul, limbaje şi tehnici de programare; • Rezolvarea problemelor practice concrete care includ elemente de programare şi utilizare de sisteme de calcul cu microprocesoare sau microcontrolere.

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

5.1 Curs	Se va desfăşura într-o sală dotată cu videoproiector şi computer, împreună cu toate accesoriile aferente: cabluri de alimentare (date şi semnal video), telecomandă videoproiector.
5.2 Seminar/ Laborator/Proiect	Se va desfăşura într-o sală cu dotare specifică, care trebuie să includă: • Sisteme de calcul de uz general (calculatoare personale), necesare implementării metodelor şi algoritmilor specifici, pe care este instalat suportul software necesar (sistem de operare şi aplicaţii de lucru specific necesar – mediu de lucru integrat de tip IDE) • Tablă albă alături de dotările specifice: marker-e speciale, burete pentru tablă albă, soluţii de curăţare a tablei. Prezenţa este obligatorie la activităţile de laborator (cerinţă conformă Regulamentelor interne de desfăşurare a activităţilor în UNSTPB).

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.)*

Disciplina este prevăzută în anul 1, semestrul 2.

Reprezintă o disciplină inginerească de tip fundamental, care oferă bazele de studiu în special pentru specializările ulterioare bazate pe lucrul intensiv cu calculatorul personal, care oferă în planul de învăţământ discipline fundamental bazate pe utilizarea/proiectarea algoritmilor şi scrierea de cod sursă (implementare).

Pentru curs se urmăreşte însuşirea conceptelor programării orientate pe obiecte necesare modelării pe calculator a diverselor probleme din viaţa reală, dar şi cele implicate de proiectarea şi implementarea unor aplicaţii dedicate sau a temelor întâlnite în practică. Sunt analizate situaţii practice variate şi sunt implementaţi algoritmi într-un limbaj de nivel înalt proiectat folosind principiile obiectuale de proiectare software (ISO C++). Programele realizate la laborator ajută la formarea reflexelor mentale în modelarea realităţii şi ajută studenţii în activitatea viitoare de inginer. Abilităţile şi deprinderile oferite de parcurgerea disciplinei constituie cunoştinţe fundamentale absolut necesare disciplinelor specifice direcţiilor de specializare orientate către discipline software, pentru care studenţii optează la trecerea în ciclul de studii următor (începând cu anul III de studiu).

Pentru laborator (aplicaţii) se propune dezvoltarea unor rutine generale, care pentru aplicaţii numerice. Astfel, se are în vedere construirea facilă de către proiectanţii hardware/software a unei biblioteci specifice. Utilizarea în conceperea programelor a limbajului ISO C/C++, asigurând astfel portabilitatea programelor şi permiţând interfaţarea şi lucrul imediat cu majoritatea pachetelor de dezvoltare/proiectare, care se bazează pe aceste limbaje. Subiectele de studiu/implementare sunt bazate pe concepte şi principii specifice

programării obiectuale, precum: clase şi obiecte, funcţii membre, constructori, destructori, ascunderea datelor, abstractizare, funcţii prietene, referinţe şi pointeri (pointerul this), moştenire simplă şi multiplă, polimorfism sau supraîncărcare.

Abilităţile deprinse în urma scrierii programelor realizate la laborator devin instrumente de lucru pentru studenţi, în activitatea desfăşurată la proiectele de an şi licenţă (diplomă) precum şi în activitatea viitoare de inginer software.

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

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 învăţării reflectă realizările studentului şi mai puţin intenţiile profesorului. Rezultatele învăţării informează studenţii despre ceea ce se aşteaptă de la ei din punct de vedere al performanţei, pentru a obţine notele şi creditele dorite. Sunt definite în termeni concreţi, folosind verbe similare exemplelor de mai jos şi indică ceea ce se va urmări prin evaluare. Rezultatele învăţării vor fi astfel redactate încât să fie evidenţiată clar relaţia faţă de competenţele definite la punctul 7.)*

Cunoştinţe	*Rezultatul asimilării de informaţii prin învăţare. Cunoştinţele reprezintă ansamblul de fapte, principii, teorii şi practici legate de un anumit domeniu de muncă sau de studiu. Pot fi teoretice şi/sau faptice.* • **Enumeră** conceptele obiectuale fundamentale care stau la baza limbajului obiectual studiat. • **Înţelege** faptul că implementarea nu este singura activitate de bază în contextul programării obiectuale şi mai general al dezvoltării software. Alături de aceasta intervin analiza cerinţelor de proiectare şi proiectarea în sine a unei aplicaţii. • **Defineşte** noţiunile UML (Unified Modelling Language) specifice domeniului. UML prezintă la nivel formal interacţiunile posibile dintre clasele din care face parte un program, asimilabilă etapei de proiectare a aplicaţiei ce urmează să fie implementată. • Este capabil să **descrie** zonele componente ale programului C++ implementat, necesar rezolvării unei probleme date (prezentată în limbaj natural).

Aptitudini	*Capacitatea de a aplica cunoştinţe şi de a utiliza know-how pentru a duce la îndeplinire sarcini şi a rezolva probleme. Aptitudinile sunt descrise ca fiind cognitive (implicând utilizarea gândirii logice, intuitive şi creative) sau practice (implicând dexteritate manuală şi utilizarea de metode, materiale, unelte şi instrumente).* • **Identifică şi descri**e conceptele obiectuale de bază împreună cu modalităţile de implementare ale acestora: ascunderea datelor/funcţiilor, reutilizarea codului, supraîncărcarea şi suprascrierea funcţiilor. • **Modelează** problemele propuse, formalizând textul problemei (exersează etapa de proiectare a programului/aplicaţiei). • **Identifică** clasele (ulterior obiectele) care interacţionează în aplicaţia ce urmează să fie scrisă. Corolar: **exersează** abilităţile de abstractizare şi translatare către limbajul de programare a ideilor exprimate 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. • T**estează, depanează şi rulează** programul scris (aplicaţia implementată) şi este capabil să îndrepte posibilele erori, identificându-le în prealabil, alături de posibilele consecinţe.
Responsabilitate şi autonomie	*Capacitatea cursantului de a aplica în mod autonom şi responsabil cunoştinţele şi aptitudinile sale.* • **Demonstrează** receptivitate pentru contexte noi de învăţare (principiile programării obiectuale constituie o nouă paradigmă, continuând paradigma structurată). • **Respectă** principiile de etică academică, înţelegând responsabilităţile asumate de rezolvare individuală a problemelor propuse, folosind metodele şi tehnicile învăţate. • **Demonstrează** autonomie în organizarea situaţiei/contextului de învăţare sau a situaţiei problemă de rezolvat • **Conştientizează** valoarea contribuţiei sale în domeniul ingineriei software, din perspectiva vieţii active, odată cu identificarea şi propunerea unor soluţii proprii, care să rezolve probleme din viaţa socială şi 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 studenţilor la stabilirea propriului parcurs de învăţare, cum se identifică eventualele rămâneri în urmă şi ce măsuri remediale se adoptă în astfel de cazuri.)*

Plecând de la analiza caracteristicilor de învăţare ale studenţilor şi de la nevoile lor specifice, identificate de către titularul de curs în urma experienţei personale în mediul privat, procesul de predare foloseşte metode de predare atât expozitive (prelegerea, expunerea), cât şi conversative-interactive, bazate pe modele de învăţare bazate pe acţiune, precum exerciţiul sau rezolvarea de probleme de programare. Interactivitatea cu studenţii prin intermediul părţii aplicative asociate conceptelor predate este în mod special evidenţiată.

Partea de modelare se reduce adesea la rezolvarea unor probleme practice, identificate în viaţa reală, prin modelare obiectuală. Se prezintă modul în care principiile programării obiectuale se reflectă imediat în modalităţile concrete de implementare (legătura existentă între principiile obiectuale şi practica programării obiectuale). Conceptele prezentate se reîntâlnesc şi în anii de specializare (anii 3 şi 4), în cadrul unor discipline precum: Deep Learning şi Inteligenţă Artificială, Robotică sau Reţele Neuronale şi Sisteme Fuzzy.

Dialogul din timpul cursului se prelungeşte şi în cadrul şedinţelor de laborator. Acestea sunt necesare pregătirii studenţilor pentru testele de verificare individuale de pe parcurs. O şedinţă tipică începe cu o scurtă trecere în revistă a noţiunilor de programare specifice lucrării (acolo unde este cazul, cu prezentarea comparativă a variantei programării structurate). Ulterior, studenţii urmăresc să conceapă şi să scrie programe complete, funcţionale.

Limbajul folosit este ISO C++, în varianta standard C++ 2011. Mediul de dezvoltare open-source folosit în laborator este configurabil în acest sens şi poate fi instruit să folosească varianta standard indicată a limbajului ISO C++ prin folosirea opţiunii de compilare -std=c++11. Mediul integrat de programare şi materialul aferent platformelor pentru laborator sunt disponibile studenţilor sub formă electronică.

Prezentările (curs + laborator) utilizează imagini şi scheme/reprezentări UML (aferente algoritmilor utilizaţi – unde este cazul, dar şi a modelării obiectuale – UML) astfel încât informaţiile prezentate să fie uşor de înţeles şi asimilat.

10. Conţinuturi

CURS		
Capitolul	**Conţinutul**	**Nr. ore**
1	Conceptele fundamentale ale programării orientată pe obiecte.	3
2	Caracterizarea C++ (prin comparaţie cu ANSI/ISO C). Exemple concrete utilizând sintaxa specifică C++.	2
3	Clase şi obiecte. Distincţia static-dinamic din punctul de vedere al efectelor unui program. Descrierea conceptelor obiectuale referitoare la clase: funcţii membre, constructori, destructori, ascunderea datelor, abstractizare.	2
4	Funcţii membre (metode): detalii funcţionale. Exemple. Specificatori de acces la date.	2
5	Constructori: detalii. Constructor implicit, cu parametru, de copiere. Exemple. Destructori. Exemple. Funcţii prietene. Exemple.	3
6	Referinţe C++. Pointerul this. Pointeri către clase. Static vs. dinamic într-un astfel de context. Exemple.	2
7	Membri de tip static ai unei clase. Funcţii statice.	2
8	Moştenire: introducere. Concepte şi reguli specifice.	3
9	Moştenire: continuare. Moştenire simplă. Specificatorii de acces la date şi metode. Moştenire de tip public, protejat sau privat.	2
10	Moştenire multiplă. Polimorfism. Supraîncărcarea funcţiilor. Legare devreme şi târzie. Metode virtuale.	3
11	Moştenirea interfeţei vs. moştenirea implementării. Supraîncărcarea operatorilor. Operatori supraîncărcabili. Tratarea excepţiilor.	2
12	Recapitulare: exemple de exerciţii şi probleme.	2
	Total:	28

Bibliografie:

1. Şl. Dr. ing. GROSU Vlad-Alexandru, PCLP2, *suport de curs electronic* (https://curs.upb.ro/2021/course/view.php?id=8979) sau POO: https://curs.upb.ro/2024/course/view.php?id=7978)
2. Brian Overland, *C++ – Ghid pentru începători*, Ed. Corint, Bucureşti, 2006.
3. Danny KALEV, Michael TOBLER, Jan WALTER – *C++*, Waite Group, January 1999 (publicat şi în Editura Teora, 2000)
4. Ionuţ MUŞLEA – *Iniţiere în C++ (Programare orientată pe obiecte)*, Ed. Microinformatica, Cluj, 1993.
5. Dan SOMNEA, Doru TURTUREA – *Iniţiere în C++ (programarea orientată pe obiecte)*, Ed. Tehnică, 1993.
6. Emanuela Cerchez, Marinel Şerban – *Programarea în limbajul C/C++ pentru liceu*, vol. 4, Ed. Polirom, ≥ 2013.

LABORATOR

Nr. crt.	Conţinutul	Nr. ore
1	Recapitulare structura unui program. Testarea aptitudinilor de programare (scurt test de verificare). Comparaţie între programarea procedurală (structurată) şi cea obiectuală.	2
2	Introducere în clase şi obiecte. Distincţia practică dintre cele două concepte.	2
3	Clase şi obiecte (cont.). Constructori: cu parametric, de copiere. Funcţii prietene.	2
4	Constructori (cont.). Destructori. Funcţii cu argument obiecte şi referinţe la obiecte.	2
5	Moştenire simplă.	2
6	Moştenire multiplă. Polimorfism.	2
7	Verificare finală laborator	2
	Total:	14

Bibliografie:

1. Şl. Dr. ing. GROSU Vlad-Alexandru, PCLP2, suport de curs electronic (Moodle) (https://archive.curs.upb.ro/2021/course/view.php?id=8979)
2. Brian OVERLAND – *C++ - Ghid pentru începători*, Ed. Corint, 2006.
3. Herbert SCHILDT – *C++ - Manual complet*, Ed. Teora, 2001.
4. Adonis BUTUFEI – Introducere în C++, InfoAcademy Net (https://pdfcoffee.com/curs-infoacademy-c-pdf-free.html)

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 și practice de aplicare a metodelor și tehnicilor predate. • Definirea conceptelor, principiilor și metodelor folosite în domeniile: programarea calculatoarelor, limbaje de nivel înalt și specifice, arhitectura sistemelor de calcul, sisteme electronice programabile, grafică, arhitecturi software reconfigurabile.	Susținerea probei individuale scrise de examinare (total: 40p). Se investighează atât capacitatea studenților de a formaliza o problemă cât și pe cea de transpunere în program a conceptelor specifice. Subiectele acoperă atât partea teoretică aferentă definirii conceptelor programării obiectuale, cât și pe cea practică, din perspectiva capacității de rezolvare cu ajutorul limbajului C++ a unor probleme de programare (prezentate în limba română).	40%
11.5 Seminar/laborator/proiect	• (C4.5) Susținerea și promovarea unei probe referitoare la arhitectura și principiile funcționale ale unei structuri software funcționale. • (C6.5) Susținerea unei probe privind stabilirea și descrierea operațiilor necesare pentru realizarea și testarea unui algoritm specific, conceput pe baza paradigmei programării orientate pe obiecte.	Activitatea de laborator este verificată constant, pe tot parcursul semestrului, prin intermediul: • testelor de activitate la fiecare laborator: 10% • testului de verificare de la jumătatea semestrului (la calculator): 20% • colocviu: 30% – laboratorul se încheie cu verificarea finală, individuală, la stație. Aceasta se bazează pe implementarea unei probleme, enunțată în limbaj natural, ce acoperă materia întregului semestru.	60%
11.6 Condiții de promovare			
• Se verifică abilitățile de acumulare și apoi de identificare a situațiilor practice specifice metodelor prezentate precum și a aplicării corecte a acestor metoda în domeniul ingineriei informaționale. • Promovarea materiei presupunea acumularea (fără praguri intermediare impuse) a ***cel puțin 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)

Universitatea Naţională de Ştiinţă şi Tehnologie Politehnica Bucureşti

Facultatea de Electronică, Telecomunicaţii şi

Tehnologia Informaţiei

În prezent se impune pregătirea viitorilor ingineri şi cercetători în domeniul algoritmilor numerici de rezolvare prin diferite metode matematice a problemelor de proiectare, testare sau prelucrare a diferitelor semnale. Formarea studenţilor pe trunchiul de programare asigură cunoştinţele fundamentale necesare în orice activitate profesională ulterioară: învăţământ, cercetare şi/sau proiectare.

Activităţile aferente Programării Calculatoarelor oferă bazele gândirii algoritmice şi ale programării într-un limbaj de programare actual, aşa cum este limbajul ISO C++ conform ultimei sale standardizări din anul 2011 (ISO/IEC 14882:2011).

Prin structurarea informaţiei conform activităţilor prevăzute în curriculum, ca şi prin activitatea de tutorat desfăşurată, materia oferă paşii necesari aprecierii calităţii, meritelor şi limitelor unor procese, programe, proiecte, concepte, metode şi teorii.

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, 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
21.09.2026	S.l./Lect. Dr. Vlad-Alexandru Grosu 	S.l./Lect. Dr. Vlad-Alexandru Grosu 

Data avizării în departament	Director de departament
22.09.2026	Prof. Dr. Claudius Dan 

Data aprobării în Consiliul Facultăţii	Decan
22.09.2026	Prof. Dr. Mihnea Udrea 