**Universitatea Națională de Știință și Tehnologie Politehnica București**

**Facultatea de Electronică, Telecomunicații și Tehnologia Informației**

# FIȘA DISCIPLINEI

## 1. Date despre program

| | |
|---|---|
| 1.1 Instituția de învățământ superior | **Universitatea Națională de Știință și Tehnologie Politehnica București** |
| 1.2 Facultatea | **Electronică, Telecomunicații și Tehnologia Informației** |
| 1.3 Departamentul | **Dispozitive, Circuite și Arhitecturi Electronice** |
| 1.4 Domeniul de studii | Inginerie Electronică, Telecomunicaţii şi Tehnologii Informaţionale |
| 1.5 Ciclul de studii | Licență |
| 1.6 Specializarea | Electronică aplicată |

## 2. Date despre disciplină

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2.1 Denumirea disciplinei (ro)<br>(en) | | | | Programarea calculatoarelor și limbaje de programare 1<br>Computer Programming and Programming Languages 1 | | | |
| 2.2 Titularul activităților de curs | | | | Conf. Dr. Radu Hobincu | | | |
| 2.3 Titularul activităților de seminar / laborator | | | | As. Drd. Costin-Emanuel VASILE, As. Drd. Alexandra-Mihaela ENESCU, As. Drd. Andrei-Alexandru ULMĂMEI, As. Drd. Nicolae GUZU | | | |
| 2.4 Anul de studiu | 1 | 2.5 Semestrul | I | 2.6. Tipul de evaluare | E | 2.7 Regimul disciplinei | Ob |
| 2.8 Tipul disciplinei | | F | 2.9 Codul disciplinei | 04.F.01.O.004 | | 2.10 Tipul de notare | Nota |

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

| | | | | | |
|---|---|---|---|---|---|
| 3.1 Număr de ore pe săptămână | 4 | Din care: 3.2 curs | 2.00 | 3.3 seminar/laborator | 2 |
| 3.4 Total ore din planul de învățământ | 56.00 | Din care: 3.5 curs | 28 | 3.6 seminar/laborator | 28 |
| Distribuția fondului de timp: | | | | | ore |
| Studiul după manual, suport de curs, bibliografie și notițe<br>Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate<br>Pregătire seminarii/ laboratoare/proiecte, teme, referate, portofolii și eseuri | | | | | 37 |
| Tutorat | | | | | 3 |
| Examinări | | | | | 4 |
| Alte activități (dacă există): | | | | | 0 |
| 3.7 Total ore studiu individual | 44.00 | | | | |
| 3.8 Total ore pe semestru | 100 | | | | |
| 3.9 Numărul de credite | 4 | | | | |

## 4. Precondiții (acolo unde este cazul)

| | |
|---|---|
| 4.1 de curriculum | • Algebră și Analiză Matematică<br>• Logică matematică |
| 4.2 de rezultate ale învățării | Aplicarea cunoștințelor, conceptelor și metodelor de bază din informatică, algebră și logica matematică |

**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 și semnal video, telecomandă). |
| 5.2 Seminar/ Laborator/Proiect | Se va desfășura într-o sală cu dotare specifică, care trebuie să includă:<br>• Sisteme de calcul de uz general (calculatoare personale), necesare implementării metodelor și algoritmilor specifici, pe care este instalat suportul software necesar (sistem de operare și aplicații de lucru specific necesar – mediu de lucru integrat de tip IDE);<br>• Tablă albă alături de dotările specifice: marker-e speciale, burete pentru tablă albă, soluții de curățare a tablei.<br><br>Prezența este obligatorie la activitățile de laborator (cerință conformă Regulamentelor interne de desfășurare a activităților în POLITEHNICA București) |

**6. Obiectiv general** *(Se referă la intențiile profesorilor pentru studenți, la ceea ce studenții vor fi învățaţi în timpul cursului. Oferă o orientare cu privire la locul cursului în cadrul domeniului științific abordat, precum și 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 învățământ al specializării studiate etc.)*

Disciplina este prevăzută în anul 1, semestrul 1 (trunchiul comun al pregătirii inginerești).

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 învățământ discipline fundamental bazate pe utilizarea/proiectarea algoritmilor, scrierea de cod sursă (implementare) precum și activităţi de depanare (identificarea erorilor semantice).

Pentru curs se urmărește însușirea conceptelor programării structurate (în perspectiva fundamentării din punct de vedere sintactic a celei *orientate pe obiecte în C++*, care va urma) necesare modelării pe calculator a diverselor probleme din viața reală, dar și cele implicate de proiectarea și implementarea unor aplicații dedicate sau a temelor întâlnite în practică. Sunt analizate situații practice variate și 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 realității și ajută studenții în activitatea viitoare de inginer. Abilitățile și deprinderile oferite de parcurgerea disciplinei constituie cunoştinţe fundamentale absolut necesare disciplinelor specifice direcţiilor de specializare orientate către discipline software, pentru care studenții optează la trecerea în ciclul de studii următor (începând cu anul III de studiu).

Pentru laborator (aplicații) se propune dezvoltarea unor rutine generale de nivel cel mult mediu, dedicate rezolvării unor situații întâlnite în viața reală. Astfel, se are în vedere construirea facilă de către proiectanții hardware/software a unei biblioteci specifice. Utilizarea în conceperea programelor a limbajului ISO C/C++, asigurând astfel portabilitatea programelor și permițând interfațarea și lucrul imediat cu majoritatea pachetelor de dezvoltare/proiectare, care se bazează pe aceste limbaje înrudite.

Abilitățile deprinse în urma scrierii programelor realizate la laborator (și a mediului de dezvoltare integrat) devin instrumente de lucru pentru studenți, mai întâi pentru activitățile din cadrul anilor de specializare (în funcție de opțiunile exprimate) iar apoi, în perspectivă, pentru activitățile desfășurate la proiectele de an și licență (diplomă) și în cele din urmă în mediul privat (viitor inginer software).

**7. Competențe** *(Capacitatea dovedită de a utiliza cunoștințe, aptitudini și abilități personale, sociale și/sau metodologice în situații de muncă sau de studiu și pentru dezvoltarea profesională și personală. Reflectă cerințele angajatorilor.)*

| | |
|---|---|
| **Specifice** | C3:Aplicarea cunoștințelor, conceptelor și metodelor de bază privitoare la arhitectura sistemelor de calcul, microprocesoare, microcontrolere, limbaje și tehnici de programare. |
| **Transversale (generale)** | CT1:Analiza metodică a problemelor întâlnite în activitate, identificând elementele pentru care există soluții consacrate, asigurând astfel îndeplinirea sarcinilor profesionale;<br>CT3: Adaptarea la noile tehnologii, dezvoltarea profesională și personală, prin formare continuă folosind surse de documentare tipărite, software specializat și resurse electronice în limba română și, cel puțin, într-o limbă de circulație internațională. |

**8. Rezultatele învățării** *(Sunt enunțuri 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.)*

| | |
|---|---|
| **Cunoștințe** | *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.*<br><br>• **Enumeră** conceptelor fundamentale care stau la baza limbajului de programare studiat.<br>• **Înțelege** faptul că implementarea nu este singura activitate de bază în contextul programării structurale/obiectuale și mai general al dezvoltării software.<br>• **Analizează** cerințele de proiectare și pașii necesari proiectării în sine a unei aplicații (consecință a enunțului anterior).<br>• **Este capabil să descrie** zonele componente ale programului ISO C 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).*<br>• **Identifică și descrie** conceptele de bază ale programării într-un limbaj de uz general (structurat, procedural) împreună cu modalitățile de implementare ale acestora: ascunderea datelor/funcțiilor, reutilizarea codului, supraîncărcarea și suprascrierea funcțiilor.<br>• **Modelează** problemele propuse, formalizând textul problemei (exersează etapa de proiectare a programului/aplicației).<br>• **Exersează** abilitățile de abstractizare și translatare către limbajul de programare a ideilor exprimate în limbaj natural.<br>• **Identifică soluții** și **elaborează** planuri de rezolvare pentru problemele propuse (exprimate în limbaj natural).<br>• **Analizează și compară** soluția găsită problemei propuse spre rezolvat cu sugestiile oferite de către titularul de laborator pentru problema specifică de rezolvat.<br>• **Testează, depanează și rulează** programul scris (aplicația implementată) și este capabil să îndrepte posibilele erori, identificându-le în prealabil, alături de posibilele consecințe (mai ales de ordin semantic). |
| **Responsabilitate și autonomie** | *Capacitatea cursantului de a aplica în mod autonom și responsabil cunoștințele și aptitudinile sale.*<br>• **Demonstrează receptivitate** pentru contexte noi de învățare (principiile programării obiectuale constituie o noutate, din perspectiva planului de învățământ).<br>• **Respectă** principiile de etică academică, înțelegând responsabilitățile asumate de rezolvare individuală a problemelor propuse, folosind metodele și tehnicile învățate.<br>• **Demonstrează autonomie** în organizarea situației/contextului de învățare sau a situației problemă de rezolvat.<br>• **Conștientizează** 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.)*

Plecând de la analiza caracteristicilor de învățare ale studenților și de la nevoile lor specifice, identificate de către titularul de curs în urma experienței personale în mediul privat, procesul de predare folosește metode de predare atât expozitive (prelegerea, expunerea), cât și conversative-interactive, bazate pe modele de învățare bazate pe acțiune, precum exercițiul sau rezolvarea de probleme de programare. Interactivitatea cu studenții prin intermediul părții aplicative asociate conceptelor predate este în mod special evidențiată.

Sunt rezervate intervale de prezentare și rezolvare a unor probleme curente pe care studenții le întâlnesc la discipline din anul 2 (fundamentale sau din domeniul electronic): modelarea dispozitivelor electronice, analiza componentelor și circuitelor pasive precum și pregătirea abilităților inginerești de rezolvare a problemelor cu ajutorul calculatorului (metode numerice).

Partea de modelare se reduce adesea la rezolvarea unor probleme practice, identificate în viața reală. Conceptele prezentate sunt cele pregătitoare disciplinelor corelate următoare din anii 1 și 2 (*Programarea Calculatoarelor și Limbaje de Programare 2*, *Structuri de Date și Algoritmi*, *Metode Numerice*) și mai târziu în anii de specializare (anii 3 și 4), în cadrul unor discipline precum: *Deep Learning și Inteligență Artificială*, *Robotică* sau *Rețele Neuronale și Sisteme Fuzzy*.

Dialogul din timpul cursului continuă în cadrul *ședințelor de laborator*. Acestea sunt necesare pregătirii studenților pentru testele de verificare individuale de pe parcurs, care formează abilitățile de rezolvare a problemelor contra-timp. O ședință tipică începe cu o scurtă trecere în revistă a noțiunilor de programare specifice lucrării. Ulterior, studenții urmăresc să conceapă și să scrie programe complete, funcționale - plecând de la enunțul în *limbaj natural* (limba română).

Limbajul folosit este ISO C, în varianta standard C 1999. Mediul de dezvoltare open-source folosit în laborator este configurabil în acest sens și poate fi instruit să folosească varianta standard indicată a limbajului ISO C prin folosirea opțiunii de compilare -std=c99. Mediul integrat de programare și materialul aferent platformelor pentru laborator sunt disponibile studenților sub formă electronică.

**10. Conținuturi**

| **CURS** | | |
|---|---|---|
| **Capitolul** | **Conținutul** | **Nr. ore** |
| 1 | = Introducere =<br>Noțiunea de compilator. Standardele C.<br>Comentariu bibliografic. | 2 |
| 2 | = Structura programelor C =<br>Distincția sintaxă-semantică. Componentele unui program. Alfabetul limbajului. Cuvintele-cheie. | 2 |
| 3 | = Memoria: zone și spații de memorie =<br>Alinierile în memorie. Baze de numerație. Tipuri de memorie. Modele de memorie folosite de către compilator. Tipurile de date fundamentale din C. Declararea și definirea variabilelor și constantelor. Clase de memorare. Variabile globale. | 2 |
| 4 | = Operatori și expresii =<br>Clasificarea operatorilor. Exemple pentru clasele de operatori identificate. Ordine de precedență și asociativitate. Conversiile implicite și explicite. Detalii asupra zonei de comentarii din structura generală a programului. Detalii asupra fișierelor antet. Detaliu asupra zonei directivelor preprocesor. Funcții inline (macro): introducere. | 2 |
| 5 | = Instrucțiuni =<br>Instrucțiunile expresie, bloc, decizie (if), selecție (switch), ciclare. Exemple.<br>Instrucțiunile de salt: continue, break, goto. | 2 |
| 6 | = Funcții (introducere) =<br>Declarație (prototip) și definiție (corp). Variabile locale și externe funcțiilor. Apel de funcții. Parametri formali și efectivi.<br>Stiva. Convenții de apel. Funcții macro (detalii). Comparația funcțiilor iterative și recursive. | 2 |
| 7 | = Funcții (continuare) =<br>Funcții inline (continuare): diferențele între macro-expandare și apelul de funcție.<br>Funcția main(): detalii. Modularizare. Detaliu asupra zonei de definire a tipurilor de date de către utilizator (typedef). Detaliu asupra zonei prototipurilor de funcții. | 2 |

| | | |
|---|---|---|
| 8 | = Pointeri =<br>Declarare, inițializare. Operatorii de preluare a adresei și de-referențiere. Valoarea specială NULL. Aritmetica pointerilor. Tipul void aplicat la pointeri. Funcții cu argument pointer și tip returnat pointer. Pointeri către funcții. Tehnica alocării dinamice de memorie. | 4 |
| 9 | = Vectori =<br>Vectori 1-dimensionali și n-dimensionali: declarare, definire. Inițializare. Indexare. Șiruri. Legătura vector-pointer. Funcții cu argument vector. Enumerări. declarare/definire. Enumerările și pointerii. Funcții cu argument enumerări. | 3 |
| 10 | = Tipuri de date neomogene =<br>Structuri: declarare, definire, inițializare. Operații permise asupra structurilor. Structurile și pointerii. Operatorul sizeof aplicat structurilor. Uniuni: declarare/definire, inițializare. | 3 |
| 11 | = Limbajul C în operații de intrare/ieșire (I/O) =<br>Noțiunile de flux și fișier. Fișiere. Operații asupra fișierelor: deschidere, închidere, scriere, citire, interogare/stabilirea valorii indicatorului de poziție. Ștergerea fizică a fișierelor. Redenumire.<br>Funcții de tratare a erorilor. | 2 |
| 12 | = Aspecte avansate ale limbajului =<br>Conceptul de variabilă. Declarații/definiții (detalii de implementare). Domeniul de vizibilitate și durata de viață a variabilelor. Legarea variabilelor. Clase de memorare și spații ale numelor. Valoare dreapta-valoare stânga. Etapele compilării. Funcții cu număr variabil de argumente. Funcții recursive (detaliu). Crearea de tipuri de date proprii: tipuri de bibliotecă. | 2 |
| | **Total:** | 28 |

**Bibliografie:**

1. Șl. dr. ing. Vlad-Alexandru Grosu - PCLP 1 - suport de curs, format electronic (platforma Moodle): https://curs.upb.ro/2023/course/view.php?id=10034
2. I. Rusu, Dana Gavrilescu, Vlad Al. Grosu - ”Programarea calculatoarelor în limbaj C”, Editura MatrixRom, București, 2002.
3. I. Rusu, Dana Gavrilescu, Vlad Al. Grosu – ”Îndrumar de laborator pentru programarea calculatoarelor: C”, Editura MatrixRom, București, 2004.
4. I. Rusu, Vlad Al. Grosu – ”Programarea calculatoarelor în limbaj C: probleme rezolvate şi comentate”, Editura MatrixRom, București, 2008.
5. D.I. Năstac, “Programarea calculatoarelor în limbajul C – Elemente fundamentale”, Editura Printech, București, 2006.
6. D.I. Năstac, “Structuri de date şi algoritmi – Aplicaţii”, Editura Printech, București, 2008.
7. D. Burileanu, C. Dan, M. Pădure, “Programare în C. Culegere de probleme”, Editura Printech, București, 2004.
8. Brian Kernighan, Dennis Ritchie – "The C programming language", Prentice Hall, New Jersey, edițiile 1978 și 1988.

**LABORATOR**

| Nr. crt. | Conținutul | Nr. ore |
|---|---|---|
| 1 | Baze de numerație. Conversii între bazele de numerație uzuale. | 2 |

| | | |
|---|---|---|
| 2 | Alfabetul limbajului. Cuvintele-cheie. Declararea și definirea variabilelor. | 2 |
| 3 | Corp de instrucțiuni. Operatori (introducere).Construcția expresiilor. Instrucțiuni. | 2 |
| 4 | Operatori (continuare): operatorul de atribuire. Conversia explicită (cast). Operatorul de secvenţiere. Definiția tipurilor utilizator: typedef. Instrucțiunea switch-case. Întreruperi și salturi în C. Ciclul for. | 2 |
| 5 | Instrucțiunile while și do-while. Vectori (introducere). | 2 |
| 6 | Vectori (continuare): șiruri. Alocarea statică.<br>Pointeri (I. Introducere). | 2 |
| 7 | Pointeri (II). Pointer NULL. Aritmetica pointerilor. Legătura vector-pointer. | 2 |
| 8 | Pointeri (III): pointerii și alocarea dinamică. | 4 |
| 9 | Structuri. Operatorul typedef aplicat structurilor. Pointerii și structurile: construcția unui tip abstract de dată. | 2 |
| 10 | Enumerări. Uniuni. Deosebiri între uniuni și structuri. | 2 |
| 11 | Operații I/O. Funcțiile de bibliotecă utilizate în lucrul cu fișierele. | 2 |
| 12 | Verificare finală laborator | 4 |
| | **Total:** | 28 |

**Bibliografie:**

1. Șl. dr. ing. Vlad-Alexandru Grosu - PCLP 1 - suport de laborator, format electronic (site personal): https://itlectures.ro/ro_RO/p-c/materiale-studiu-pc/
2. I. Rusu, Dana Gavrilescu, Vlad Al. Grosu - ”Programarea calculatoarelor în limbaj C”, Editura MatrixRom, București, 2002.
3. I. Rusu, Dana Gavrilescu, Vlad Al. Grosu – ”Îndrumar de laborator pentru programarea calculatoarelor: C”, Editura MatrixRom, București, 2004.
4. I. Rusu, Vlad Al. Grosu – ”Programarea calculatoarelor în limbaj C: probleme rezolvate și comentate”, Editura MatrixRom, București, 2008.
5. Brian Kernighan, Dennis Ritchie – "The C programming language", Prentice Hall, New Jersey, ediţiile 1978 și 1988.
6. Herbert Schildt – "C – manual complet", Editura Teora, 1999-2003.
7. Florin Munteanu, Gh. Muscă, Florin Moraru – "C - tehnici de programare", Editura Joint Printing House, București, 1995.

**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.<br>- Aprofundarea noțiunilor de analiză matematică și algebră necesare cursului, cu aplicare în domeniul electronicii (domeniu de pregătire fundamentală).<br>- C4.1: 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. | - Se investighează atât capacitatea studenților de a formaliza o problemă cât și pe cea de transpunere în program a conceptelor specifice.<br>- Subiectele acoperă atât partea teoretică aferentă definirii conceptelor programării structurate, cât și pe cea practică, din perspectiva capacității de rezolvare cu ajutorul limbajului ISO C a unor probleme de programare concrete (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.<br>- C6.5: susținerea unei probe privind stabilirea și descrierea operațiilor necesare pentru realizarea și testarea unui algoritm specific, conceput pe baza paradigmei programării structurate. | Activitatea de laborator este verificată constant, pe tot parcursul semestrului, prin intermediul:<br>- testelor de activitate la fiecare laborator: 10p<br>- testului de verificare de la jumătatea semestrului (la calculator): 20p<br>- colocviu: 30p.<br>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: 30p. | 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.<br>• Promovarea materiei presupunea acumularea (fără praguri intermediare impuse) a **cel puțin 50p** din totalul de 100p aferent. | | | |

**12. Coroborarea conținutului disciplinei cu așteptările reprezentanților angajatorilor și asociațiilor profesionale reprezentative din domeniul aferent programului, precum și cu stadiul actual al cunoașterii în domeniul științific abordat și practicile în instituții de învățământ superior din Spațiul European al Învățământului Superior (SEÎS)**

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

Activitățile aferente disciplinei *Programarea Calculatoarelor și Limbaje de Programare 1* oferă bazele gândirii algoritmice și ale programării într-un limbaj de programare actual, așa cum este limbajul ANSI/ISO C conform standardizării din anul 1999 (ISO/IEC 9899:1999).

Prin structurarea informației conform activităților prevăzute în curriculum, ca și prin activitatea de tutorat desfășurată, materia oferă pașii necesari aprecierii calității, meritelor și limitelor unor procese, programe, proiecte, concepte, metode și teorii.

Utilizarea adecvată a criteriilor și metodelor de evaluare, conforme normelor academice europene la care universitatea POLITEHNICA București este parte, permite studenților să se autoevalueze continuu, pornind de la calificativele obținute dar și ținând cont de observațiile și indicațiile metodologice pe care titularul de curs/laborator le oferă.

| Data completării | Titular de curs | Titular(i) de aplicații |
|---|---|---|
| 21.02.2025 | Conf. Dr. Radu HOBINCU | As. Drd. Costin-Emanuel VASILE |
| | | As. Drd. Alexandra-Mihaela ENESCU |
| | | As. Drd. Andrei-Alexandru ULMĂMEI |
| | | As. Drd. Nicolae GUZU |

| Data avizării în departament | Director de departament |
|---|---|
| | Prof. Dr. Claudius DAN |

Data aprobării în Consiliul Facultății Decan

Prof. Dr. Radu-Mihnea UDREA