# Programa școlară
pentru disciplina

# *Informatică*

## Clasa a X-a

Curriculum de specialitate (CS)

*Filiera teoretică, profilul real, specializarea științe ale naturii*

Învățământ liceal

- 2026 –

**Anexa nr. 53**
**Programa școlară pentru disciplina *Informatică,* CS, clasa a X-a**

- *Filiera teoretică, profilul real, specializarea științe ale naturii*

# NOTĂ DE PREZENTARE

***Statutul disciplinei***

Disciplina *informatică*, studiată în ciclul liceal, este o continuare a disciplinei *informatică și TIC*, studiată în gimnaziu, în trunchiul comun. În ciclul liceal, aceasta are rolul de a consolida și extinde domeniile de competență, aprofundând aspectele privind conceptuale, algoritmice și aplicative ale domeniului informatic.

Conform Ordinului ministrului educației și cercetării nr. 4.350/2025 privind aprobarea planurilor-cadru pentru învățământul liceal cu frecvență zi, disciplina *informatică* se studiază ca disciplină din categoria curriculumului de specialitate (CS) la:

- filiera teoretică, profilul real, specializarea matematică-informatică, în clasele a IX-a, a X-a, a XI-a și a XII-a;
- filiera teoretică, profilul real, specializarea științe ale naturii, în clasele a IX-a și a X-a;
- filiera vocațională, profilul militar, specializarea matematică-informatică militară, în clasele a IX-a, a X-a, a XI-a și a XII-a.

Pentru filiera teoretică, profilul real, specializarea științe ale naturii, alocarea orară este:

- clasa a IX-a – 1 oră/săptămână, pentru studiu teoretic și activități practice;
- clasa a X-a – 1 oră/săptămână, pentru studiu teoretic și activități practice.

Activitățile practice sunt desfășurate obligatoriu în laboratorul de informatică.

**Raportarea la cadrul legislativ și documentele strategice generale și specifice care susțin/întemeiază studiul disciplinei**

Elaborarea prezentei programe școlare este bazată pe documente fundamentale care definesc viziunea și structura curriculumului național:

- Legea învățământului preuniversitar nr. 198/2023, cu modificările și completările ulterioare, precum și alte acte subsecvente relevante privind implementarea curriculumului național;
- Ordinul privind aprobarea planurilor-cadru pentru învățământul liceal cu frecvență zi (Ordinului ministrului educației și cercetării nr. 4.350/2025);
- Recomandarea Consiliului UE privind competențele-cheie pentru învățarea pe tot parcursul vieții (2018);
- Cadrul european al calificărilor (EQF);
- Rapoarte OECD/UNESCO privind competențele digitale și educația STEM, Cadre de referință;
- Profilul de formare al absolventului (Ordinul ministrului educației 6731/2023);
- Cadrul european al competențelor digitale pentru cetățeni (DigComp), respectiv Cadrul de Competențe digitale pentru elevi DigiComp 2.2. (Anexa OME 6466/2024).

**Rolul disciplinei în formarea elevilor**

Studiul disciplinei *informatică* vizează formarea unei gândiri computaționale, algoritmice, analitice și creative, capabile să abordeze probleme complexe prin modele și instrumente specifice științei calculatoarelor, valorificând competențele formate pe parcursul ciclului gimnazial. Informatica se bazează pe o gândire riguroasă, o capacitate de abstracție și modelare, dar și pe utilizarea tehnologiilor digitale în contexte variate — academice, profesionale și cotidiene.

Disciplina contribuie direct la realizarea profilului de formare al absolventului de liceu, dezvoltând competențe-cheie definite la nivel european, cum ar fi în principal, competența digitală, competența matematică și competența în științe, tehnologie și inginerie, dar, indirect, și celelalte competențe cheie europene, prin activități de învățare adecvate și utilizarea limbajului de specialitate în diferite contexte.

„Ideile mari" promovate de disciplină vizează dezvoltarea gândirii computaționale, rezolvarea problemelor de natură informatică, elaborarea unor algoritmi eficienți, scrierea codului clar și funcțional, analiza complexității soluțiilor propuse, precum și interacțiunea cu diferite modele conceptuale de organizare a datelor.

Utilitatea studiului disciplinei *informatică* se manifestă pe multiple planuri:

- pe parcursul școlar, prin fundamentarea gândirii computaționale, necesare și pentru alte discipline;
- în viața cotidiană, prin planificarea riguroasă a acțiunilor, utilizarea critică și creativă a instrumentelor digitale;
- în formarea profesională, prin deschiderea către cariere în IT, automatizare, robotică, analiză de date, securitate cibernetică sau inteligență artificială.

**Justificarea statutului disciplinei *informatică,* elemente de continuitate/de noutate**

Disciplina *informatică* valorifică achizițiile formate în gimnaziu la nivelul competențelor digitale de bază și al utilizării instrumentelor informatice, aducând progres prin abordarea formală și științifică a modelelor de organizare a datelor, a algoritmilor specializați pe tipuri de probleme, programarea structurată și orientată pe obiecte, eficiența rezolvărilor, precum și elemente de inteligență artificială și baze de date.

Caracterul său de curriculum de specialitate (CS) subliniază rolul central al disciplinei în formarea competențelor profesionale ale elevilor în domeniul informatic. Prin natura sa, informatica se află la intersecția între cunoaștere teoretică și aplicare practică, consolidând gândirea logică și deschiderea către știință, inovație și responsabilitate digitală.

**Categorii de programe școlare pentru disciplina *informatică***

Disciplina *informatică* se încadrează în categoria Curriculumului de specialitate (CS), fiind destinată aprofundării competențelor specifice.

Conform planurilor cadru, în învățământul liceal, componenta fixă a curriculumului de specialitate **poate fi completată cu o componentă flexibilă**, ce include discipline/module de specialitate și alocări orare la decizia unității de învățământ; curriculumul de specialitate poate fi sprijinit/completat cu opționale de aprofundare, noi discipline sau poate fi valorizat din perspectivă integrată/interdisciplinară, prin oferta națională de opționale sau prin oferta unității de învățământ.

La acesta se adaugă **curriculumul la decizia elevului din oferta școlii** (CDEOȘ), prin opționale care pot fi de aprofundare, noi discipline/module de pregătire sau opționale integrate prin care elevii pot primi sprijin pentru remediere, consolidare sau performanță, pentru pregătirea suplimentară, respectiv pentru pregătirea susținerii atestatelor de certificare a competențelor profesionale sau pentru satisfacerea unor nevoi și interese pentru specializare la nivel superior, inclusiv pentru admiterea la studii universitare.

**Orientări generale și specifice în lectura programei școlare**

Aplicarea programei presupune:

- respectarea competențelor generale și specifice ca repere obligatorii în proiectarea activităților didactice;
- proiectarea didactică flexibilă, centrată pe situații de învățare autentice și evaluări bazate pe competențe;
- corelarea competențelor generale și specifice cu domeniile de conținut și cu exemplele de activități de învățare;
- planificarea integrată a conținuturilor, cu accent pe rezolvarea de probleme, lucrul în echipă, proiecte interdisciplinare și utilizarea resurselor digitale moderne;
- valorificarea componentelor orientative pentru adaptarea la particularitățile colectivului de elevi și la resursele școlii;
- utilizarea Python ca limbaj de programare de bază și a limbajelor C++, respectiv a SQL, conform programei școlare corespunzătoare profilului, clasei și specializării;
- utilizarea unor resurse digitale moderne în laboratoare de informatică dotate adecvat;
- corelarea activităților de învățare cu domenii STEM, economie, științe sociale și artă digitală.

Programa are caracter obligatoriu în ceea ce privește competențele generale, competențele specifice și conținuturile precizate, iar componentele metodologice și exemplele de activități de învățare au caracter orientativ, oferind cadrul pentru adaptarea la nivelul clasei și al resurselor disponibile. Profesorul are libertatea de a selecta și combina metode, resurse și instrumente digitale adecvate, cu condiția respectării finalităților și competențelor prevăzute de programă. Se recomandă accentuarea caracterului practic, formativ și explorator al disciplinei, pentru a consolida autonomia elevului în învățare și în formarea unei culturi informatice autentice.

Alegerea limbajului Python ca limbaj de programare de bază în studiul disciplinei *informatică* răspunde cerințelor unui învățământ modern, centrat pe formarea gândirii algoritmice și a competențelor digitale relevante. Datorită sintaxei sale simple și clare, Python facilitează înțelegerea conceptelor fundamentale de programare, permițând elevilor să se concentreze pe rezolvarea problemelor și pe dezvoltarea logicii algoritmice. Totodată, prin caracterul său actual și aplicativ, Python este utilizat pe scară largă în domenii precum analiza datelor, inteligența artificială, automatizare și dezvoltare software, oferind elevilor o pregătire relevantă pentru continuarea studiilor și integrarea profesională.

Studierea limbajului C++ are o valoare formativă și permite aprofundarea conceptelor fundamentale de programare și a gândirii algoritmice. După însușirea principiilor de bază în Python, C++ oferă elevilor oportunitatea de a înțelege mai profund mecanismele interne ale programării, precum gestionarea memoriei și structurile de date dinamice.

Introducerea noțiunilor de învățare automată (Machine Learning) în programa școlară corespunzătoare clasei, profilului și specializării, este importantă pentru familiarizarea elevilor cu una dintre cele mai importante ramuri ale informaticii moderne. Studiul învățării automate permite elevilor să înțeleagă cum pot fi antrenate modelele și algoritmii pentru a recunoaște tipare și a realiza predicții pe baza datelor, folosind limbajul Python și biblioteci specifice. Prin activități practice, precum clasificarea imaginilor, analiza datelor sau predicția unor valori, elevii dobândesc competențe reale de analiză și programare aplicată. Învățarea automată contribuie astfel la dezvoltarea gândirii logice și algoritmice, oferind o bază solidă pentru studii superioare și cariere în domenii precum informatică, știința datelor sau inteligența artificială.

Sistemele informatice moderne (site-uri web, aplicații, platforme educaționale) depind de baze de date pentru a funcționa automatizat. Studiul bazelor de date este important în formarea competențelor digitale, deoarece acestea reprezintă fundamentul gestionării eficiente a datelor în orice domeniu de activitate. Într-o societate bazată pe date, capacitatea de a colecta, organiza, analiza și interpreta informații este esențială pentru viața profesională și personală.

Setul de competențe generale ale disciplinei *informatică* este construit pe baza taxonomiei Bloom (revizuite), urmărind o dezvoltare progresivă a gândirii logice și algoritmice, de la nivelurile de înțelegere și aplicare până la cele de analiză, evaluare și creare. Această structură sprijină elevii în formarea unei viziuni integrate asupra procesului de dezvoltare software, de la concept la implementare, asigurând totodată experiență de învățare în domeniul informaticii.

Prin această abordare, competențele generale facilitează o evoluție cognitivă clară: de la identificarea și explicarea modelelor conceptuale și operaționale care stau la baza programării, la aplicarea și analiza acestora în contexte variate, continuând cu evaluarea critică a soluțiilor informatice și crearea de produse software originale, adaptate cerințelor.

În ansamblu, competențele generale precizate în programă contribuie la formarea competențelor digitale, logice și creative necesare, ca bază, unui specialist în domeniul informatic, într-o societate bazată pe tehnologie și inovare.

Programa școlară de *informatică* este calibrată astfel încât să asigure o rezervă de 25% din timpul alocat disciplinei, la dispoziția cadrului didactic pentru activități de remediere, consolidare, aprofundare sau extindere.

Structura programei școlare include, pe lângă Nota de prezentare, următoarele componente:

- Competențe generale;
- Competențe specifice și exemple de activități de învățare;
- Conținuturi;
- Sugestii metodologice.

**Competențele generale (CG)** vizate la nivelul disciplinei integrează achizițiile de cunoaștere și de comportament așteptate, subliniind orientarea generală a procesului educațional la această disciplină.

Competențele generale sunt derivate din competențele-cheie și explicitează finalitățile majore ale disciplinei, acele achiziții de durată pe care toți elevii trebuie să le dobândească prin întreg studiul acesteia, la nivelul ciclului liceal. Acestea dau coerență disciplinei, stabilesc direcția învățării și fundamentează derivarea competențelor specifice, selecția și organizarea conținuturilor învățării. Competențele generale au grad ridicat de complexitate și integrează ansambluri de cunoștințe, abilități și atitudini, ca rezultate ale învățării utile pentru dezvoltarea personală, pentru cetățenia activă, pentru incluziune socială și pentru angajare pe piața muncii.

**Competențele specifice (CS)** sunt competențe derivate din competențele generale și reprezintă etape măsurabile în formarea și dezvoltarea acestora, ilustrând rezultate ale învățării pentru fiecare an de studiu. Acestea exprimă, pentru elevi, achizițiile învățării prin parcurgerea disciplinei de studiu, și includ, la fel ca în cazul competențelor generale, ansambluri de cunoștințe, abilități și atitudini. Competențele specifice asigură continuitatea de la gimnaziu, progresia de la un an la altul și conexiunea cu profilul de formare al absolventului.

Pentru formarea și dezvoltarea competențelor specifice, în programă sunt propuse **exemple de activități de învățare (EAI)**, care descriu contexte și modalități prin care competențele specifice sunt formate, exersate, consolidate și evaluate în mod curent, formativ. Ele au rol orientativ, nu prescriptiv, și oferă profesorilor repere privind modul în care pot organiza situații de învățare relevante pentru elevi. Astfel, profesorul poate să adapteze activitățile de învățare propuse în programă, să le completeze sau să le înlocuiască cu altele adecvate clasei, asigurând cadrul unui demers didactic personalizat, pentru formarea/dezvoltarea competențelor prevăzute de programă, în contextul specific al fiecărei clase.

**Conținuturile** sunt organizate în domenii de conținut (categorii mari) și, în cadrul acestora, în conținuturi propriu-zise ale învățării. Domeniile de conținut și conținuturile învățării definesc „substanța” disciplinei: ce anume se studiază efectiv, pentru a sprijini formarea competențelor. Acestea constituie o selecție, adecvată din punctul de vedere didactic, de elemente din domeniul de studiu al disciplinei (informații factuale, conceptuale, procedurale), cu rol de suport operațional/instrumental pentru formarea competențelor specifice. Selecția este făcută pe baza principiului continuității și al coerenței, iar conținuturile sunt interconectate, astfel încât, după parcurgerea lor integrală, elevul să fie capabil să realizeze conexiuni, în scopul rezolvării unor probleme diverse, de natură teoretică sau practic-aplicativă.

**Sugestiile metodologice** au rolul de a sprijini profesorii în aplicarea programei, fără a impune sau a face o inventariere a metodelor didactice utilizate. Acestea traduc intențiile programei (competențe generale, competențe specifice, conținuturi, exemple de activități de învățare) în modalități și mijloace pentru realizarea demersului didactic, prin exemple minimale, relevante, de abordare a activității didactice, pentru alegerea strategiilor didactice și pentru integrarea conținuturilor și competențelor în practica școlară.

Astfel, programa școlară oferă un cadru coerent de utilizare: competențele generale indică direcțiile de învățare, competențele specifice, pe ani de studiu, precizează etapele de progres, domeniile de conținut stabilesc suportul științific pentru formarea acestor competențe, iar exemplele de activități de învățare ilustrează modalități concrete de dezvoltare a experiențelor de învățare.

## COMPETENȚE GENERALE (CG)

| | |
|---|---|
| **CG1** | **Identifică principalele caracteristici ale modelelor conceptuale și operaționale ale dezvoltării produselor software, pentru înțelegerea fundamentelor programării** |
| **CG2** | **Explică principii care stau la baza modelelor conceptuale și operaționale ale dezvoltării produselor software, pentru a fundamenta în mod logic proiectarea și implementarea soluțiilor informatice** |
| **CG3** | **Utilizează modele conceptuale și operaționale ale dezvoltării produselor software, în scopul obținerii de soluții informatice funcționale și eficiente** |
| **CG4** | **Analizează caracteristicile și aplicabilitatea modelelor conceptuale și operaționale ale dezvoltării produselor software, pentru a selecta soluțiile cele mai potrivite în funcție de contextul informatic dat** |
| **CG5** | **Evaluează corectitudinea și eficiența soluțiilor informatice, în vederea optimizării și asigurării funcționalității în diverse scenarii de utilizare** |
| **CG6** | **Elaborează algoritmi și programe personalizate, pentru a crea soluții informatice coerente și adaptate cerințelor** |

# COMPETENȚE SPECIFICE (CS)
# ȘI
# EXEMPLE DE ACTIVITĂȚI DE ÎNVĂȚARE (EAI)

**CG 1 - Identifică principalele caracteristici ale modelelor conceptuale și operaționale ale dezvoltării produselor software, pentru înțelegerea fundamentelor programării**

| **Clasa a X-a** |
|---|
| **CS 1.1. Identifică principalele caracteristici ale organizării datelor în cadrul unor modele conceptuale simple - liniare, asociative, mixte, pentru a structura și accesa date în vederea prelucrării acestora** |
| - *definirea conceptului de șir de caractere ca o colecție ordonată și imutabilă de caractere, prin analogie cu textul unui eseu sau cu un mesaj scris pe telefon*<br>- *enumerarea operațiilor de bază asupra șirurilor de caractere (acces la un caracter sau la o secvență de caractere, concatenare, comparare lexicografică, căutare) în contextul verificării sau construirii unui nou mesaj pe baza unui text trimis de un elev către colegii săi pentru organizarea unei activități școlare*<br>- *identificarea caracteristicilor unui model mixt, de tipul listă de liste, pentru memorarea mediilor elevilor dintr-o clasă, în care fiecare listă corespunde câte unui elev și conține mediile pentru fiecare disciplină, astfel încât elementul aflat în lista i, pe poziția j, reprezintă media elevului i la disciplina j* |
| **CS 1.2. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea listelor** |
| - *recunoașterea caracteristicilor metodei căutării binare cu specificarea condiției necesare, în contextul localizării rapide a unui contact într-o agendă telefonică ordonată alfabetic*<br>- *enumerarea etapelor metodei căutării binare în contextul localizării unui titlu într-o listă ordonată de cărți dintr-o bibliotecă digitală*<br>- *recunoașterea caracteristicilor metodei interclasării în contextul combinării a două liste ordonate de produse provenite din depozite diferite pentru a obține o singură listă finală, ordonată*<br>- *enumerarea pașilor necesari aplicării metodei interclasării în contextul unificării a două liste ordonate de participanți înscriși online și offline la un eveniment, astfel încât rezultatul să fie o listă unică și ordonată* |
| **CS 1.3. Identifică specificul, caracteristicile și etapele unor strategii de tip Greedy pentru rezolvarea problemelor** |
| - *definirea caracteristicilor metodei Greedy în contextul alegerii celei mai bune opțiuni la fiecare pas, atunci când un elev își planifică trei activități dintr-o listă în care sunt asociate duratele activităților, astfel încât să maximizeze timpul liber*<br>- *recunoașterea caracteristicilor algoritmilor care utilizează metoda Greedy, în contexte aplicative, precum selectarea celor mai avantajoase oferte într-o aplicație de cumpărături, în care deciziile sunt luate secvențial, fără posibilitatea revenirii asupra alegerilor anterioare* |
| **CS 1.4. Identifică principalele elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele simple - liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice** |
| - *identificarea modalităților de reprezentare a șirurilor de caractere în Python (apostrofuri simple, triple și ghilimele) prin asocierea cu modurile în care putem cita sau nota o expresie în vorbirea curentă: între apostrofuri, ghilimele simple sau ghilimele/apostrofuri triple în contexte cotidiene ca notarea unui titlu, a unui citat sau a unui text mai lung (paragraf, textul unui e-mail etc.)*<br>- *recunoașterea unei modalități de accesare a unui caracter dintr-un șir în contextul utilizării unei litere dintr-un cuvânt, la o anumită poziție cu evidențierea faptului că, în Python, caracterele dintr-un șir pot fi accesate prin indexare, așa cum putem număra pozițiile literelor dintr-un cuvânt*<br>- *identificarea modului de accesare a unui caracter dintr-un șir de caractere folosind operatorul de indexare, în contextul afișării primei litere dintr-un nume introdus într-un formular digital*<br>- *descrierea corespondenței dintre caracterele unui șir și valorile lor ASCII în contextul codificării unui text prin înlocuirea fiecărui caracter cu codul său ASCII reprezentat prin trei cifre*<br>- *identificarea elementelor de sintaxă ale limbajului Python pentru inițializarea unei variabile corespunzătoare memorării datelor organizate sub forma unui model mixt, de tipul listă de liste, pentru memorarea relațiilor de vecinătate în cadrul unui grup de țări, în care fiecare listă corespunde câte unei țări, astfel încât elementul aflat în lista i, pe poziția j, este 1, dacă țările i și j sunt vecine și este 0, altfel*<br>- *identificarea elementelor de sintaxă ale limbajului C++ pentru inițializarea unei variabile de tip tablou bidimensional, corespunzătoare memorării datelor organizate sub forma unui model mixt, de tipul listă de liste, pentru memorarea relațiilor de prietenie în cadrul unui grup de persoane, în care fiecare listă corespunde câte unei persoane, astfel încât elementul aflat în lista i, pe poziția j, este 1, dacă persoanele i și j sunt prietene și este 0, altfel* |

**CG 2 - Explică principii care stau la baza modelelor conceptuale și operaționale ale dezvoltării produselor software, pentru a fundamenta în mod logic proiectarea și implementarea soluțiilor informatice**

| **Clasa a X-a** |
|---|
| **CS 2.1. Explică principiile de organizare și prelucrare a datelor în cadrul unor modele conceptuale simple - liniare, asociative, mixte, pentru a le gestiona eficient** |
| - *explicarea modului în care este prelucrat un șir de caractere, în contextul funcționării unui sistem de verificare a complexității parolelor, care analizează fiecare simbol pentru a identifica litere mari, litere mici, cifre și caractere speciale*<br>- *explicarea modului în care concatenarea și separarea șirurilor de caractere sunt utilizate pentru combinarea sau divizarea informațiilor textuale în contextul generării automate a etichetelor pentru colete, unde numele destinatarului și adresa sunt reunite sau separate în funcție de necesitățile sistemului*<br>- *explicarea modului de acces la date în cazul unui model mixt, de tipul listă de liste, pentru memorarea mediilor elevilor dintr-o clasă, în care fiecare listă corespunde câte unui elev și conține mediile pentru fiecare disciplină, astfel încât elementul aflat în lista i, pe poziția j, reprezintă media elevului i la disciplina j* |
| **CS 2.2. Explică etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea listelor** |
| - *explicarea etapelor metodei căutării binare în contextul găsirii rapide a unui contact într-o agendă telefonică ordonată alfabetic, evidențiind faptul că numele căutat este localizat prin împărțirea repetată a listei în jumătăți succesive*<br>- *explicarea importanței ordonării elementelor într-o listă, arătând că doar o listă sortată permite utilizarea eficientă a căutării binare (de exemplu, pentru a identifica rapid un zbor într-o listă ordonată după ora plecării) și a interclasării (de exemplu, pentru a combina două liste ordonate de rezervări ale unei săli într-un registru unic, fără suprapuneri sau erori)*<br>- *explicarea etapelor metodei de interclasare, arătând cum două liste ordonate sunt combinate într-una singură prin compararea elementelor lor pe rând, folosind ca exemplu situația unui hotel care primește rezervări din două surse diferite (formular online și agenție de turism), fiecare sursă generând propria listă ordonată, iar hotelul având nevoie să le reunească într-o singură listă cu rezervări ordonate cronologic* |
| **CS 2.3. Explică etapele unor strategii de tip Greedy pentru rezolvarea problemelor** |
| - *explicarea principiului metodei Greedy pentru rezolvarea unei situații concrete, de exemplu, selectarea, pentru un muncitor, a unui număr maxim de sarcini dintr-o listă de activități, care trebuie rezolvate într-un timp dat, prin alegerea repetată a sarcinii care consumă cel mai puțin timp disponibil*<br>- *explicarea necesității demonstrării corectitudinii utilizării metodei Greedy comparând situații în care această metodă poate conduce uneori la rezultate greșite, de exemplu, pentru problema formării unei sume de bani date cu minimizarea numărului de monede necesare, atunci când sistemul de monede nu este „canonic” (de exemplu, pentru suma 6 și setul de monede 1, 3, 4 metoda Greedy oferă un rezultat necorespunzător, cu trei monede, adică 4+1+1, în loc de soluția de două monede, adică 3+3)*<br>- *explicarea etapelor metodei Greedy aplicate unei probleme practice de planificare, precum selectarea unui număr maxim de spectacole care se pot desfășura în aceeași sală, într-o zi, fără suprapuneri, cunoscându-se intervalele de desfășurare a spectacolelor* |
| **CS 2.4. Explică rolul principalelor elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele simple - liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice** |
| - *explicarea efectului aplicării metodelor uzuale ale clasei str, în contextul ilustrării în limbaj de programare a unor exemple inspirate din viața cotidiană, de exemplu, separarea cuvintelor dintr-o propoziție (split()), transformarea textului în litere mici sau mari pentru uniformizare (lower(), upper()), eliminarea spațiilor nedorite de la începutul sau sfârșitul unui mesaj (strip()), înlocuirea unui cuvânt greșit (replace()), numărarea aparițiilor unui caracter într-un text (count()), verificarea faptului că un text conține doar litere (isalpha()) sau doar cifre (isdigit())*<br>- *explicarea utilizării operatorului de indexare pentru accesarea unui element corespunzător unei variabile Python în care sunt memorate date organizate sub forma unui model mixt, de tipul listă de liste, pentru memorarea relațiilor de vecinătate în cadrul unui grup de țări, în care fiecare listă corespunde câte unei țări, astfel încât elementul aflat în lista i, pe poziția j, este 1, dacă țările i și j sunt vecine și este 0, altfel*<br>- *explicarea utilizării operatorului de indexare pentru accesarea unui element corespunzător unei variabile C++ de tip tablou bidimensional, în care sunt memorate date organizate sub forma unui model mixt, de tipul listă de liste, pentru memorarea relațiilor de prietenie în cadrul unui grup de persoane, în care fiecare listă corespunde câte unei persoane, astfel încât elementul aflat în lista i, pe poziția j, este 1, dacă persoanele i și j sunt prietene și este 0, altfel* |

**CG 3 - Utilizează modele conceptuale și operaționale ale dezvoltării produselor software, în scopul obținerii de soluții informatice funcționale și eficiente**

| **Clasa a X-a** |
|---|
| **CS 3.1. Utilizează modul de organizare și prelucrare a datelor în cadrul unor modele conceptuale simple - liniare, mixte, pentru a rezolva probleme de gestionare a datelor** |
| - *utilizarea comparării lexicografice pentru ordonarea titlurilor de cărți într-o bibliotecă digitală, astfel încât acestea să fie afișate alfabetic într-o listă*<br>- *aplicarea operațiilor specifice de parcurgere a unui șir de caractere pentru a verifica dacă acesta reprezintă un număr de telefon (conține exclusiv cifre și are exact 10 caractere)* |

| Clasa a X-a |
|---|
| - *aplicarea operației de concatenare a șirurilor de caractere pentru a construi un mesaj complet, folosind date separate precum numele unui elev și media acestuia, într-un text personalizat*<br>- *utilizarea unui model mixt, de tipul listă de liste, pentru memorarea mediilor elevilor dintr-o clasă, în care fiecare listă corespunde câte unui elev și conține mediile pentru fiecare disciplină, astfel încât elementul aflat în lista i, pe poziția j, reprezintă media elevului i la disciplina j* |
| **CS 3.2. Aplică algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea listelor** |
| - *aplicarea metodei de căutare binară pentru găsirea unui element într-o listă ordonată, folosind ca exemple căutarea unui număr de telefon într-o agendă electronică în care contactele sunt aranjate în ordine crescătoare după numărul de telefon*<br>- *aplicarea metodei de căutare binară pentru o listă dată de programări dintr-o zi, ordonate cronologic, pentru a verifica dacă se poate face o programare pentru ora „11:00”, condiția fiind ca la acea oră să nu existe deja o programare*<br>- *aplicarea metodei de căutare binară prin urmărirea pas cu pas a valorilor limitelor de căutare și a poziției elementului verificat, pentru un set dat de valori ordonate, până la identificarea elementului căutat sau stabilirea faptului că acesta nu există în listă, în contextul unei liste ordonate de coduri ale unor produse*<br>- *utilizarea metodei interclasării pentru combinarea a două liste date de produse provenind din două depozite, produsele din fiecare listă fiind ordonate crescător în funcție de preț, obținându-se lista finală cu toate produsele ordonată crescător în funcție de preț* |
| **CS 3.3. Aplică strategii de tip Greedy pentru rezolvarea problemelor** |
| - *aplicarea metodei Divide et impera pentru determinarea sumei necesare pentru plata unor produse, dându-se o listă care conține prețurile lor, prin împărțirea repetată a acesteia și a fiecărei liste obținute pe parcurs în jumătăți (până când se ajunge la liste formate dintr-o singură valoare, pentru care se cunoaște rezultatul) și determinarea rezultatului pentru o astfel de listă, prin adunarea sumelor corespunzătoare celor două jumătăți, după obținerea acestora*<br>- *aplicarea metodei Divide et impera în rezolvarea unor probleme clasice, precum mutarea obiectelor între suporturi respectând reguli precise (problema turnurilor din Hanoi), determinarea valorii maxime sau minime dintr-o listă de date sau localizarea rapidă a unei informații într-o evidență ordonată (căutarea binară)*<br>- *aplicarea metodei de sortare prin interclasare pentru obținerea unei liste ordonate pornind de la o listă neordonată formată din 7 numere date*<br>- *aplicarea metodei Greedy pentru rezolvarea unei situații concrete, de exemplu, selectarea pentru o anumită zi și un timp dat, a unui număr maxim de sarcini dintr-o listă de activități, alegând de fiecare dată sarcina care consumă cel mai puțin timp*<br>- *aplicarea metodei Greedy pentru rezolvarea unei probleme practice, precum selectarea dintr-o listă de spectacole candidate a se desfășura într-o sală, a unui număr maxim de spectacole ce pot fi programate fără a exista suprapuneri* |
| **CS 3.4. Utilizează principalele elemente ale limbajului de programare pentru prelucrarea datelor organizate în modele simple - liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice** |
| - *utilizarea metodelor clasei str în diferite situații de procesare a textului în contexte cotidiene cum ar fi formatarea automată a numelor participanților la un concurs după un model dat (lower(), upper(), title(), capitalize()), validarea unui nume prin verificarea faptului că textul introdus conține doar litere (isalpha()), verificarea dacă un cod introdus conține doar cifre (isdigit())*<br>- *utilizarea metodelor clasei str în diferite situații de modificare a textului în contexte cotidiene cum ar fi curățarea datelor introduse de utilizatori într-un formular online prin eliminarea spațiilor suplimentare (strip()), separarea cuvintelor dintr-o propoziție pentru o prelucrare ulterioară a acestora (split())*<br>- *utilizarea operatorului de indexare pentru accesarea unui element corespunzător unei variabile Python în care sunt memorate date organizate sub forma unui model mixt, de tipul listă de liste, pentru memorarea relațiilor de vecinătate în cadrul unui grup de țări, în care fiecare listă corespunde câte unei țări, astfel încât elementul aflat în lista i, pe poziția j, este 1, dacă țările i și j sunt vecine și este 0, altfel*<br>- *utilizarea operatorului de indexare pentru accesarea unui element corespunzător unei variabile C++ de tip tablou bidimensional, în care sunt memorate date organizate sub forma unui model mixt, de tipul listă de liste, pentru memorarea relațiilor de prietenie în cadrul unui grup de persoane, în care fiecare listă corespunde câte unei persoane, astfel încât elementul aflat în lista i, pe poziția j, este 1, dacă persoanele i și j sunt prietene și este 0, altfel* |

**CG 4 - Analizează caracteristicile și aplicabilitatea modelelor conceptuale și operaționale ale dezvoltării produselor software, pentru a selecta soluțiile cele mai potrivite în funcție de contextul informatic dat**

| Clasa a X-a |
|---|
| **CS 4.1. Analizează caracteristicile, modul de prelucrare, avantajele și dezavantajele utilizării unor modele conceptuale simple - liniare, asociative, mixte, pentru a alege structura adecvată în rezolvarea unor probleme concrete** |
| - *compararea diferitelor modalități de prelucrare a unui șir de caractere care conține numele urmat de toate prenumele unei persoane, prin separarea acestor componente pentru a le reuni într-un nume format din toate prenumele urmate de nume*<br>- *analizarea modelelor de organizare a datelor care să permită prelucrarea unui mesaj real, precum un e-mail sau un anunț online, pentru a separa informațiile esențiale (expeditorul, destinatarul, subiectul și mesajul principal)* |

| Clasa a X-a |
|---|
| - *compararea a două moduri de stocare a datelor despre produse – într-o listă de liste (unde poziția contează) și într-o listă de dicționare (unde accesul se face prin chei), pentru a alege varianta mai clară și mai flexibilă* |
| **CS 4.2. Analizează comportamentul, aplicabilitatea, avantajele și dezavantajele utilizării unor algoritmi specializați pe clase de probleme pentru prelucrarea listelor** |
| - *examinarea modului în care metoda căutării binare restrânge treptat intervalul de căutare într-o listă ordonată, prin raportare la situații reale precum găsirea unui zbor într-o listă de plecări ordonată după oră sau localizarea unui elev într-un registru ordonat după numărul matricol*<br>- *compararea modului de funcționare și a numărului de pași necesari în căutarea secvențială față de căutarea binară, pentru a evidenția diferențele de eficiență în funcție de dimensiunea listei*<br>- *analizarea situațiilor în care metoda căutării binare nu este utilizată adecvat, prin observarea greșelilor frecvente precum aplicarea metodei în cazul unor elemente neordonate, condiții greșite sau actualizarea incorectă a limitelor de căutare*<br>- *compararea numărului de operații necesare pentru a combina listele de rezervări provenite de la două hoteluri ale aceluiași lanț, fiecare listă fiind ordonată după același criteriu, de exemplu, data sosirii clienților, pentru a obține o singură listă finală, cu toți clienții, ordonată cronologic după momentul sosirii, prin metoda interclasării, respectiv prin concatenarea celor două liste și ordonarea listei obținute*<br>- *analizarea erorilor care pot apărea în aplicarea metodei interclasării, precum omiterea unor elemente din una dintre liste, includerea aceluiași element de mai multe ori sau pierderea ordinii atunci când elementele nu sunt comparate corect după criteriul stabilit* |
| **CS 4.3. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de tip Greedy pentru rezolvarea problemelor** |
| - *analizarea modului în care aplicarea succesivă a deciziilor locale din metoda Greedy conduce la obținerea unei soluții finale concrete, de exemplu, selectarea unui set de activități dintr-o mulțime de activități dată, care să nu se suprapună în timp, astfel încât numărul total de activități alese să fie cât mai mare, evidențiind legătura dintre fiecare alegere intermediară și rezultatul final obținut*<br>- *analizarea modului în care metoda Greedy construiește o soluție pas cu pas într-un context real, precum alegerea unor produse (care pot fi preluate și parțial) într-un buget limitat, urmărind cum fiecare alegere locală (de exemplu, selectarea produsului cu cel mai bun raport preț–valoare la acel moment) conduce la obținerea unei liste finale de produse selectate, pentru care se poate calcula explicit valoarea totală obținută, reprezentând rezultatul urmărit al aplicării metodei*<br>- *analizarea situațiilor reale în care metoda Greedy conduce la un rezultat necorespunzător, cum ar fi oferirea restului cu monede de valori diferite, evidențiind cazurile în care alegerea repetată a celei mai mari monede disponibile nu conduce la o soluție* |
| **CS 4.4. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor elemente ale limbajului de programare, pentru a identifica soluția adecvată prelucrării datelor organizate în modele simple - liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice** |
| - *analizarea avantajelor utilizării unei liste de șiruri de caractere, comparativ cu un șir compact, pentru afișarea în ordine alfabetică a unor cuvinte*<br>- *analizarea contextelor de prelucrare a șirurilor de caractere, de exemplu, pentru verificarea și corectarea mesajelor trimise într-o conversație online sau într-un document școlar, unde pot apărea greșeli cauzate de folosirea incorectă a ghilimelelor/apostrofurilor la definirea unui șir, utilizarea unor metode (split(), replace(), strip()) pentru date care nu sunt de tip str, confuzia dintre șiruri de caractere și numere în timpul procesării unui text sau încercarea de a modifica un șir, deși acesta este imutabil*<br>- *analizarea avantajelor bordării pentru memorarea datelor în Python, organizate sub forma unui model mixt, de tipul listă de liste, pentru determinarea numărului de elemente nenule care au doar vecini nuli*<br>- *analizarea necesarului de memorie utilizată de un tablou bidimensional în C++ de dimensiuni mari, pentru memorarea datelor organizate sub forma unui model mixt, de tipul listă de liste, pentru reprezentarea unui labirint în care zidurile sunt marcate cu 1, iar culoarele cu 0* |

**CG 5 - Evaluează corectitudinea și eficiența soluțiilor informatice, în vederea optimizării și asigurării funcționalității în diverse scenarii de utilizare**

| Clasa a X-a |
|---|
| **CS 5.1. Argumentează alegerea unor modele conceptuale simple - liniare, asociative, mixte și a modurilor de prelucrare a acestora pentru rezolvarea unor probleme informatice concrete** |
| - *evaluarea eficienței unei proceduri de extragere a informațiilor din fișiere text/documente, precum preluarea automată a numelor, codurilor sau valorilor dintr-un fișier, justificând dacă metoda utilizată reduce timpul de procesare și minimizează erorile față de o prelucrare manuală*<br>- *argumentarea adecvării utilizării șirurilor de caractere pentru reprezentarea și prelucrarea informațiilor textuale (de exemplu, coduri de identificare, adrese de e-mail sau nume de utilizatori), cu justificarea alegerii acestui tip de date prin avantajele legate de lizibilitate, posibilitatea validării caracter cu caracter și identificarea clară a informațiilor* |

| Clasa a X-a |
|---|
| - *evaluarea corectitudinii datelor într-un model conceptual mixt (de exemplu, listă de liste) pentru un sistem de rezervări care verifică dacă toate listele de locuri disponibile corespund cu realitatea, detectând eventualele inconsistențe (de exemplu, locuri duble sau lipsă)*<br>- *argumentarea alegerii unui model mixt pentru un sistem de catalog electronic (de exemplu, listă de liste) pentru a organiza datele despre clase, elevi și note, asigurând o structură clară și flexibilă* |
| **CS 5.2. Evaluează corectitudinea și eficiența soluțiilor cu algoritmi specializați pe clase de probleme pentru prelucrarea listelor** |
| - *evaluarea corectitudinii unei soluții în care sunt căutați, în mod repetat, elevii dintr-o listă ordonată alfabetic, justificând eliminarea căutărilor inutile în una dintre jumătățile listei*<br>- *evaluarea eficienței integrării metodei căutării binare într-o aplicație funcțională, prin justificarea modului în care aceasta permite localizarea rapidă a unei informații într-o listă ordonată*<br>- *evaluarea eficienței algoritmului de interclasare a listelor ordonate în contexte reale, precum afișarea tuturor produselor unui magazin online, în ordine crescătoare după preț, pe baza a două astfel de liste, una cu produse alimentare, și alta cu produse nealimentare, comparativ cu concatenarea celor două liste și ordonarea celei obținute* |
| **CS 5.3. Evaluează corectitudinea și eficiența algoritmilor care utilizează strategii de tip Greedy pentru rezolvarea problemelor** |
| - *evaluarea modului în care optimul local în metoda Greedy conduce la o soluție eficientă în situații practice (de exemplu, selectarea produselor cu cel mai mare raport valoare/preț într-un buget limitat), prin justificarea faptului că ordonarea datelor și selecția pas cu pas a optimului local conduc la o soluție viabilă*<br>- *evaluarea obținerii soluției optime pentru probleme rezolvate cu metoda Greedy (de exemplu, organizarea unui program de întâlniri fără suprapuneri), prin justificarea corectitudinii condițiilor alese*<br>- *argumentarea deciziei de aplicare a metodei Greedy în contexte matematice (de exemplu, realizarea unei sume maxime cu n termeni de forma $a_i*b_j$ unde $a_i \in A$, $b_j \in B$, iar A și B sunt mulțimi de numere întregi, $card(A)=n$ și $card(B) \geq n$), cu justificarea eficienței obținute ca urmare a luării deciziilor locale și a respectării condițiilor de aplicabilitate ale metodei* |
| **CS 5.4. Evaluează corectitudinea și eficiența aplicațiilor care utilizează elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele simple - liniare, mixte, în vederea rezolvării eficiente a problemelor informatice** |
| - *evaluarea eficienței și clarității redactării a două soluții pentru aceeași problemă, care verifică dacă un cuvânt este anagramă a unui alt cuvânt*<br>- *argumentarea alegerii unei metode din clasa str pentru o sarcină practică, de exemplu, căutarea unor vocale într-un text dat*<br>- *evaluarea corectitudinii expresiilor Python utilizate pentru accesarea vecinilor unui element în cazul memorării datelor organizate sub forma unui model mixt, de tipul listă de liste, în contextul unei fotografii alb-negru în care punctele corespunzătoare obiectelor sunt marcate cu 1, iar punctele de fundal cu 0*<br>- *evaluarea corectitudinii inițializării datelor organizate sub forma unui model mixt, de tipul listă de liste, pentru modelarea unei hărți în care zonele de uscat sunt marcate cu 1, iar cele de apă cu 0* |

**CG 6 - Elaborează algoritmi și programe personalizate, pentru a crea soluții informatice coerente și adaptate cerințelor**

| Clasa a X-a |
|---|
| **CS 6.1. Proiectează algoritmi personalizați care utilizează modele conceptuale simple - liniare, asociative, mixte, pentru organizarea și prelucrarea eficientă a datelor, utilizând algoritmi de bază, în vederea rezolvării unor probleme** |
| - *proiectarea unui algoritm care validează datele de tip text introduse într-un formular (nume, CNP, adresă de e-mail), prin definirea și aplicarea unor reguli clare, precum lungimea minimă și maximă a șirurilor de caractere, tipul caracterelor permise, structura obligatorie a datelor și verificarea formatului corect, cu afișarea de mesaje de eroare personalizate atunci când regulile nu sunt respectate*<br>- *crearea unui scenariu de analiză automată a textelor educaționale, care parcurge șirurile de caractere dintr-un eseu pentru a calcula frecvența cuvintelor și a genera feedback privind diversitatea vocabularului utilizat*<br>- *proiectarea unui algoritm care utilizează un model mixt (listă de liste) pentru un sistem de evidență a opțiunilor angajaților unei companii, în care există o ofertă de 10 beneficii și fiecare angajat poate opta pentru maximum cinci dintre acestea, cu posibilitatea afișării angajaților care au optat pentru un anumit beneficiu, a beneficiilor pentru care a optat fiecare angajat* |
| **CS 6.2. Proiectează soluții cu aplicarea personalizată a algoritmilor specializați pe clase de probleme pentru prelucrarea listelor** |
| - *crearea unei aplicații funcționale care integrează căutarea binară, utilizată pentru gestionarea unor date reale (de exemplu, identificarea rapidă a datelor de contact ale unui client într-o listă ordonată a acestora sau a unui produs într-o listă ordonată de inventar), evidențiind modul în care ordonarea datelor permite căutarea eficientă*<br>- *proiectarea unor soluții informatice care utilizează interclasarea pentru situații din viața reală, precum gestionarea programărilor medicale realizate independent online, respectiv telefonic, prin combinarea și accesarea eficientă a datelor*<br>- *proiectarea unor algoritmi pentru prelucrarea listelor ordonate, aplicați în contexte reale precum combinarea listelor de produse din mai multe depozite, fiecare ordonată alfabetic, prin utilizarea metodei de interclasare pentru a obține o listă finală ordonată alfabetic* |

| Clasa a X-a |
|---|
| **CS 6.3. Proiectează algoritmi cu aplicarea personalizată a strategiilor de tip Greedy pentru rezolvarea problemelor** |
| - *proiectarea unui algoritm cu metoda Greedy pentru obținerea unei soluții finale concrete, de exemplu, selectarea unui set de activități dintr-o mulțime de activități dată, care nu se suprapun în timp, astfel încât numărul total de activități alese să fie cât mai mare, prin stabilirea unui criteriu de alegere locală și aplicarea succesivă a acestuia pentru obținerea unei soluții finale*<br>- *proiectarea unui algoritm cu metoda Greedy pentru un context real, precum alegerea unor produse într-un buget limitat, urmărind cum fiecare alegere locală (de exemplu, selectarea produsului cu cel mai bun raport preț–valoare la acel moment, cu posibilitatea preluării parțiale a acestuia) conduce la obținerea unei liste finale de produse selectate, pentru care se poate calcula explicit valoarea totală obținută, reprezentând rezultatul urmărit al aplicării metodei* |
| **CS 6.4. Implementează aplicații care integrează în mod personalizat elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele simple - liniare, asociative, mixte, în vederea realizării unor soluții funcționale și performante** |
| - *implementarea unor programe personalizate folosind metode adecvate din clasa Python str pentru prelucrarea șirurilor de caractere, pentru codificarea unui text cu o regulă de criptare simplă (de exemplu, prin înlocuirea fiecărei litere cu următoarea literă din alfabet, cu excepția literei 'z', care se înlocuiește cu litera 'a')*<br>- *implementarea unor programe personalizate folosind funcționalitățile clasei str din Python, pentru verificarea formatului unei adrese de e-mail (dacă adresa conține exact un caracter '@' și cel puțin un caracter '.' după acesta, dar nu conține spații)*<br>- *implementarea unor programe Python în care se memorează date organizate sub forma unui model mixt, de tipul listă de liste, corespunzătoare unei table de șah, cu toate piesele regine, în care căsuțele cu piese albe sunt marcate cu 1, căsuțele cu piese negre sunt marcate cu -1, iar căsuțele libere sunt marcate cu 0, pentru determinarea numărului de piese atacate de o regină de altă culoare, aflată la o poziție dată*<br>- *implementarea unor programe C++ în care se memorează, într-un tablou bidimensional, date organizate sub forma unui model mixt, de tipul listă de liste, corespunzătoare unui covor de formă dreptunghiulară, în care fiecare element are o valoare ce depinde de culoarea zonei corespunzătoare, pentru a verifica simetria stânga-dreapta a modelului de pe covor* |

# CONȚINUTURI ALE ÎNVĂȚĂRII

**Clasa a X-a**

| Domenii de conținut | Conținuturi |
|---|---|
| 1. Organizarea conceptuală a datelor | 1.1. Modelul conceptual liniar – șir de caractere<br>- caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora.<br>1.2. Modelul conceptual mixt (de exemplu, liste de liste, liste de șiruri de caractere)<br>- caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază sau specifici în scopul prelucrării acestora (de exemplu, actualizarea datelor, modificarea numărului de componente ale colecției, construirea colecției). |
| 2. Strategii de rezolvare a problemelor | 2.1. Metode de prelucrare a listelor sortate<br>- caracteristici ale metodei căutării binare și repere pentru aplicarea acesteia;<br>- caracteristici ale metodei interclasării și repere pentru aplicarea acesteia.<br>2.2. Metoda Greedy<br>- caracteristici ale metodei Greedy;<br>- repere pentru aplicarea metodei. |
| 3. Memorarea datelor și organizarea codului în limbaj de programare | 3.1. Clasa str din Python – clasă predefinită pentru șiruri de caractere<br>- caracteristici;<br>- operații și operatori specifici: apartenență, concatenare, multiplicare, acces la un caracter sau subșir/secvență de caractere, comparare lexicografică;<br>- metode uzuale pentru căutare a unei secvențe de caractere, generare a unui șir prin înlocuire a unei secvențe de caractere, separare a unor secvențe de caractere dintr-un șir;<br>- repere pentru identificarea și utilizarea altor metode și funcții adecvate pentru prelucrarea șirurilor de caractere, în funcție de nevoile proprii. |

# SUGESTII METODOLOGICE

Sugestiile metodologice au rolul de a sprijini profesorii în aplicarea programei, fără a impune metode unice sau rigide. Acestea traduc intențiile programei (competențe generale, competențe specifice, conținuturi, exemple de activități de învățare) în moduri concrete de lucru la clasă și oferă repere pentru organizarea învățării și privind evaluarea rezultatelor învățării, pentru alegerea strategiilor didactice și pentru integrarea conținuturilor și competențelor în practica școlară. Această secțiune are caracter aplicativ, nu teoretic: nu inventariază metode, strategii sau instrumente, ci oferă exemple minimale, relevante.

Disciplina *informatică* are atât caracter teoretic, cât și aplicativ, iar activitățile din cadrul instruirii teoretice se desfășoară, de regulă, în săli de clasă, dotate cu tablă interactivă, pentru exemplificarea programelor pe calculator, iar activitățile din cadrul instruirii practice se desfășoară în laboratorul de informatică, unde este indicat ca fiecare elev să dispună de un calculator propriu, conectat la rețea și la Internet, pentru a facilita formarea competențelor prevăzute în programă. Stațiile de lucru trebuie să fie configurate astfel încât să permită executarea aplicațiilor specifice, iar calculatoarele să fie plasate în formă de U sau cu o orientare către tabla principală, pentru o vizibilitate optimă.
Principiile generale care trebuie să guverneze activitatea de predare-învățare-evaluare cuprind:

- centrare pe elev: strategiile să favorizeze implicarea activă a elevilor, de exemplu, învățarea prin descoperire, colaborarea și reflecția personală;
- diversitate metodologică: se recomandă utilizarea de metode variate;
- flexibilitate: profesorii adaptează activitățile de învățare la nivelul clasei și la resursele disponibile;
- corelare cu profilul de formare al absolventului: metodele didactice trebuie alese astfel încât să contribuie la formarea competențelor-cheie și a atributelor prioritare ale absolventului;
- integrare interdisciplinară: învățarea devine mai relevantă atunci când disciplinele se sprijină reciproc și creează punți între conținuturi;
- îmbinarea evaluării formative cu cea sumativă, cu recomandarea unor strategii de evaluare centrate pe o reflecție profundă asupra întrebărilor esențiale precum: De ce evaluez? Ce evaluez? Cum evaluez? Cât de bine măsor? Ce feedback dau? Ce decizii iau?;
- diferențiere/personalizare: adaptarea parcursului didactic la situații specifice (de exemplu, elevi cu CES și/ sau dizabilități, elevi cu ritm înalt de învățare, elevi care au nevoie de învățare remedială, elevi în risc de abandon etc.).

Orientările metodologice generale cuprind:

- învățare activă: dezbateri, studii de caz, proiecte, portofolii, simulări, investigații;
- învățare colaborativă: activități în grup, peer-to-peer, mentorat între elevi;
- învățare prin proiect: integrarea conținuturilor disciplinei în teme mai largi (sociale, științifice, culturale);
- învățare cu suport digital: utilizarea resurselor online, aplicații interactive, simulări virtuale;
- învățare autentică: activități conectate cu realitatea cotidiană și cu problemele comunității;
- învățare contextualizată: activități corelate cu specificul clasei/școlii în care se desfășoară procesul didactic.

Se recomandă adaptări metodologice după tipul de programă, de exemplu, pentru disciplinele din curriculumul de specialitate:

- accent pe elemente de specializare, pe aprofundare, pe rezolvarea de probleme complexe și pe gândire critică;
- metode exploratorii, investigații, cercetări aplicate;
- exemple: proiecte interdisciplinare, aplicații în domenii profesionale, utilizarea software-urilor specializate.

Formarea competențelor trebuie să aibă în vedere și legătura cu profilul de formare al absolventului, astfel încât disciplina să contribuie la dezvoltarea/consolidarea/diversificarea:

- competențelor-cheie (de exemplu, competențe matematice, digitale, sociale și civice, a învăța să înveți);
- atributelor prioritare ale profilului absolventului (de exemplu, reflexiv, creativ, responsabil, comunicativ);
- temelor transversale prevăzute de Legea 198/2023, art. 88 alin. 10 (de exemplu, educație pentru mediu, digitalizare, sănătate, patrimoniu).

Activitățile de învățare alese trebuie să urmărească formarea competențelor din programă, precum și a unor competențe transversale, cum ar fi utilizarea tehnologiilor digitale pentru a aduce îmbunătățiri sau soluții noi pentru procese și produse, cu o abordare centrată pe factorul uman, prin implicarea individuală și colectivă în procesele de gândire critică și în utilizarea creativă și intenționată a tehnologiilor digitale, pentru a înțelege și a rezolva problemele conceptuale și situațiile problematice.
De asemenea, este necesar ca elevii să fie conștienți că trebuie să rămână informați cu privire la evoluțiile tehnologice digitale și la implicațiile lor în viața personală, profesională și în societate, să recunoască domeniile în care propria competență digitală trebuie îmbunătățită sau actualizată și să abordeze propriile nevoi în materie de competențe digitale în cadrul unui proces mai amplu de învățare pe tot parcursul vieții, de consolidare a capacităților și a autonomiei, oferind deschidere spre a îi sprijini și pe alții în dezvoltarea competențelor lor digitale.
Activitățile pe calculator sunt coordonate de profesor, care definește clar sarcinile, timpul alocat și criteriile de evaluare, adaptând nivelul de dificultate în funcție de particularitățile colectivului de elevi. Activitățile de învățare trebuie să fie alese adecvat, pentru a contribui la formarea competențelor specifice din programă, astfel încât pentru nivelurile cognitive de recunoaștere și înțelegere

se recomandă activități demonstrative și exerciții de identificare, pentru nivelul de aplicare se recomandă activități practice și aplicații asistate digital, pentru nivelurile de analiză și evaluare se recomandă studii de caz și proiecte interdisciplinare, iar pentru nivelul de creare se recomandă activități de tip învățare prin acțiune, realizarea de produse digitale și proiecte în echipă.

Se recomandă îmbinarea metodelor didactice tradiționale de predare-învățare-evaluare (de exemplu, demonstrația, problematizarea, algoritmizarea, proba practică) cu cele moderne (de exemplu, învățarea prin descoperire, proiectul, portofoliul), pentru a stimula gândirea computațională și autonomia elevilor în rezolvarea de probleme informatice, profesorul având un rol preponderent de îndrumare a elevilor, în loc de unul de furnizor de informații. Abordarea exploratorie, prin testare și corectare (trial and error) sprijină formarea gândirii critice și a capacității de autoînvățare.

Mijloacele de învățământ utilizate pot fi variate, beneficiind de tehnologiile moderne care facilitează învățarea, cum ar fi aplicații specializate, software-uri educaționale, simulatoare ale comportamentului datelor prelucrate de algoritmi, tutoriale, resurse online, platforme care permit evaluarea automată a soluțiilor informatice.

Evaluarea în cadrul disciplinei informatică trebuie să aibă un caracter formativ/pe parcurs, urmărind nu doar obținerea unui rezultat corect, ci și modul în care elevii își formează gândirea algoritmică, formulează strategii, își testează algoritmii și corectează propriile erori. Se recomandă evaluări sumative/finale, după fiecare unitate de învățare.

Programa disciplinei informatică are în vedere conținuturi grupate în domenii specifice, de exemplu:
**1. Modelele conceptuale de organizare a datelor**, care vizează reprezentări abstracte ale modului în care sunt structurate și relaționate datele într-un sistem informatic — înainte ca ele să fie memorate efectiv printr-un program sau o bază de date. Ele răspund la întrebarea: „Ce informații trebuie să prelucrez/stochez și cum sunt legate între ele?” și nu la întrebarea „Ce tip de variabile utilizez pentru a le stoca concret în memorie?”. Pe parcursul studiului disciplinei în ciclul liceal sunt studiate progresiv, din punctul de vedere al complexității relațiilor dintre date, modele conceptuale fundamentale - date simple sau liste, modele conceptuale simple - liniare, neliniare sau asociative, modele conceptuale complexe – liniare, rețea sau ierarhice, respectiv modele conceptuale avansate - pentru proiectarea bazelor de date sau învățare automată.
**2. Strategii pentru rezolvarea problemelor**, care cuprind:
- algoritmi specializați pe clase de probleme, cum ar fi pentru prelucrarea algoritmică a numerelor, sortarea sau generarea sistematică a unor secvențe de valori, pentru prelucrarea listelor ordonate, criptarea sau decriptarea șirurilor de caractere, pentru prelucrarea grafurilor, arborilor, algoritmi utilizați în învățarea automată;
- strategii generale de programare/abordare cum ar fi proiectarea modulară a algoritmilor, metodele de programare Greedy, Divide et impera, Backtracking și Programare dinamică, normalizare a modelului conceptual al unei probleme de gestiune.
**3. Elemente ale limbajului de programare** pentru memorarea și prelucrarea datelor organizate în diferite modele, respectiv pentru organizarea codului (subprograme, programare orientată pe obiecte) și prelucrarea bazelor de date.

**Pentru fiecare clasă, tematica precizată trebuie abordată într-o ordine logică, facilitând formarea competențelor specifice din programă, utilizând competențele și conținuturile studiate anterior, începând cu elementele de limbaj și algoritmii de bază (pentru numărare, sumă, produs, minim, maxim, prima sau ultima valoare cu o anumită proprietate, verificarea unor proprietăți) studiați la gimnaziu.**
**Debutul poate fi făcut, după caz, cu strategiile generale de rezolvare a problemelor, conform programei, dacă acestea pot fi aplicate în continuare, împreună cu celelalte conținuturi din cadrul anului de studiu (de exemplu, metoda Backtracking, respectiv metoda Programării dinamice sunt utilizate pentru implementarea unor algoritmi specifici grafurilor).**
**Pentru prelucrarea datelor organizate sub forma diferitelor modele conceptuale, se recomandă mai întâi prezentarea unor concepte de bază privind acest tip de organizare, apoi elemente de limbaj care să sprijine memorarea datelor astfel organizate, urmate de aplicarea algoritmilor de bază pentru prelucrare, respectiv de metodele/algoritmii specializați pe clase de probleme pentru prelucrarea unor astfel de date (de exemplu, concepte de bază privind modelul conceptual liniar, apoi clasa list, urmată de aplicarea algoritmilor de bază pentru prelucrarea listelor, respectiv de metodele de sortare a unei liste).**

La rezolvarea **fiecărei probleme** de natură algoritmică, pe parcursul studiului disciplinei, se evidențiază aspecte specifice etapelor de elaborare a programelor:
- analiză (identificare a datelor de intrare și de ieșire, organizare conceptuală a datelor, descompunere a unei probleme în subprobleme, identificare a unor modele repetitive, abstractizare);
- proiectare (descompunere în module, reprezentare schematică, pseudocod);
- implementare (scriere a codului în limbaj de programare);
- testare (criterii de elaborare a testelor pentru validarea datelor și pentru verificarea comportamentului programului, urmărire a evoluției valorilor variabilelor pentru identificarea și tratarea eventualelor erori).
În parcurgerea conținuturilor se recomandă rezolvarea unor probleme concrete, întâlnite în viața reală, pentru a evidenția succesiunea etapelor de elaborare a unui program și pentru a dezvolta gândirea algoritmică a elevilor. Profesorul poate alterna reprezentările grafice, textuale și codificate ale aceluiași algoritm pentru a facilita înțelegerea legăturii dintre abstractizare și implementare.

**Un exemplu, orientativ, de ordine de abordare a conținuturilor pentru clasa a X-a, specializarea științe ale naturii:**
1. *Strategii de rezolvare a problemelor. Metode de prelucrare a listelor ordonate*
2. *Organizarea conceptuală a datelor. Modelul conceptual liniar – șir de caractere*
3. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa str din Python*

4. *Organizarea conceptuală a datelor. Modelul conceptual mixt*
5. *Strategii de rezolvare a problemelor. Metoda Greedy*

Pentru prezentarea algoritmului de căutare binară profesorul poate pune accentul pe înțelegerea principiului de împărțire repetată a domeniului de căutare, implementarea algoritmului în Python și compararea căutării binare cu cea secvențială, utilizând metode didactice adecvate, ca problematizarea și descoperirea dirijată. Printre activitățile practice se poate prezenta o simulare vizuală pe tablă sau în aplicații online, astfel încât elevii să identifice mai ușor aspectele specifice metodei.
Se recomandă proiectarea algoritmului și reprezentarea acestuia ca schiță/în pseudocod, descrierea detaliată a etapelor rezolvării unei probleme din punctul de vedere algoritmic și apoi implementarea în limbaj de programare. Pentru a identifica elementele de eficiență caracteristice acestei metode se recomandă compararea unor algoritmi de căutare secvențială cu algoritmul de căutare binară, în scopul evidențierii eficienței căutării binare.
Pentru algoritmul de interclasare este necesar ca elevii să dețină deja competențe de bază privind parcurgerea și manipularea listelor simple și utilizarea indicilor. Interclasarea poate fi prezentată ca o metodă de combinare a două liste ordonate într-o listă unică, tot ordonată, nu ca o metodă de ordonare. Profesorul pune accentul pe logica algoritmică și pe relațiile de ordine dintre elementele listelor, evitând memorarea mecanică a pașilor: clarificarea condiției de aplicare (ambele liste să fie ordonate), stabilirea modului de parcurgere sincronă (utilizarea a doi indici care cresc independent), asigurarea completării finale (după ce una dintre liste s-a epuizat, elementele rămase din cealaltă se adaugă automat în lista finală), stabilitatea metodei (interclasarea păstrează ordinea relativă a elementelor egale - concept important pentru sortările stabile), ordinul de complexitate al algoritmului (liniar în raport cu numărul total de elemente O(n + m)).
Predarea șirurilor de caractere trebuie să urmărească trecerea de la conceptul abstract de „succesiune" la manipularea concretă a caracterelor. Inițial, conceptul de șir se poate introduce fără clasa str, la nivel de abstractizare logică: o succesiune de caractere ce pot fi accesate secvențial pornind de la observații intuitive („un cuvânt e un șir de litere"). Profesorul poate utiliza demonstrația explicativă, combinată cu descoperirea asistată – elevii analizează fragmente de cod Python ce ilustrează accesul la un caracter, concatenarea, căutarea unei secvențe de caractere. Metodologic, se recomandă învățarea colaborativă și feedbackul formativ, folosind instrumente digitale precum *Python Tutor* pentru vizualizarea pas cu pas a execuției codului. După înțelegerea conceptului, se trece la implementarea în Python cu str, pentru a exersa operațiile specifice (acces, concatenare, apartenență, căutare). Această etapă consolidează gândirea secvențială și logica parcurgerii.
Predarea modelelor mixte de organizare a datelor (liste de liste, liste de șiruri de caractere etc.) poate fi abordată gradual, pornind de la exemple simple și concrete (liste de liste), în care apare nevoia de ierarhizare și acces multiplu la date. Se recomandă folosirea situațiilor reale (tabele, cataloage, colecții de obiecte) pentru a evidenția avantajele fiecărei structuri. Pentru modelul conceptual mixt de tip listă de liste se recomandă abordarea unor exemple din lumea reală, care să sugereze valori care depind simultan de două repere, modele vizuale bidimensionale. Activitățile practice și exercițiile de transformare între structuri ajută elevii să înțeleagă logica organizării datelor și să dezvolte gândirea algoritmică. Se recomandă utilizarea metodelor investigative și a cooperării pe roluri: elevii lucrează în echipe, fiecare având o sarcină (implementare, comentare, testare). Instrumentele vizuale (tabele, diagrame, editoare grafice de cod) sprijină înțelegerea relațiilor ierarhice dintre structuri.
Pentru metoda Greedy, profesorul poate utiliza învățarea prin descoperire controlată, solicitând elevilor să deducă regula de selecție optimă din exemple concrete. Este utilă strategia exemplu–contraexemplu, pentru a arăta că metoda nu este universal aplicabilă. Prin discuții metacognitive, elevii învață să argumenteze alegerile algoritmice și să compare abordări. Având în vedere că „algoritmii Greedy se folosesc de obicei în probleme de optimizare în care trebuie făcute un număr de alegeri pentru a ajunge la o soluție optimă" (Cormen, T.H., Leiserson, C.E., Rivest, R.R., *Introducere în algoritmi*), se pune accentul pe probleme de determinare a unor valori optime. Se recomandă abordarea unor rezolvări cu aplicarea algoritmilor de bază (de exemplu, sume, minime), probleme clasice (de exemplu, problema spectacolelor, problema fixării unor scânduri cu număr minim de cuie, problema rucsacului în forma continuă) ca model, apoi realizarea unor probleme asemănătoare care presupun adaptări, reinterpretări, extinderi.
Proiectele interdisciplinare și învățarea pe baza unor aplicații care să conecteze teoria cu lumea reală cresc motivația și favorizează formarea competențelor transferabile. Activitățile bazate pe învățare colaborativă încurajează comunicarea și îi pregătesc pe elevi pentru lucrul în echipă, iar repartizarea diferitelor roluri în echipă (pentru proiectarea algoritmului, implementarea programului și a subprogramelor, testarea programului) pot constitui o alfabetizare în profesiile din domeniu.
Se recomandă ca, la finalul clasei a X-a, profesorii să discute și despre perspectivele legate de competențele specifice programării, încurajând elevii să utilizeze competențele dobândite în cadrul unor proiecte de portofoliu, pentru admitere la facultate sau angajare. De exemplu, produsul software realizat la clasă poate fi perfecționat și prezentat la un concurs sau interviu. Acest tip de conexiune cu lumea reală poate evidenția eficiența curriculară, prin dimensiunea formării profesionale.

Pentru exersarea și implementarea algoritmilor în limbaje de programare elevii pot utiliza **medii de dezvoltare (IDE)**, cum ar fi cele de mai jos.
**Pentru Python:**

- PyCharm (https://www.jetbrains.com/pycharm/), variantele Community Edition și Educational Edition - multiplatformă (Windows, Linux, Mac-OS etc.);
- IDLE Python (https://www.python.org/downloads/) - multiplatformă (Windows, Linux, Mac-OS etc.);
- Spyder (https://docs.spyder-ide.org/index.html) - multiplatformă (Windows, Linux, MacOS etc.);
- Wing Wing, varianta Wing Personal (https://www.wingware.com/downloads/) - multiplatformă (Windows, Linux, MacOS etc.);
- IDE Komodo (https://github.com/Komodo/KomodoEdit) - pentru dezvoltarea aplicațiilor web și mobile, multiplatformă (Windows, Linux, MacOS etc.).

**Pentru C++:**

- Code Blocks (https://www.codeblocks.org/) - multiplatformă (Windows, Linux, MacOS etc.).

**Pentru Python și C++:**

- Visual Studio Code (https://vscode.dev) - multiplatformă (Windows, Linux, MacOS etc.).

**Instrumente online pentru implementarea soluțiilor**

- Online GDB (https://www.onlinegdb.com);
- Programiz (https://www.programiz.com/);
- w3schools (https://www.w3schools.com/);
- Repl.it (https://replit.com/) - inclusiv pentru activități de programare colaborativă;
- Jupyter Notebooks (https://jupyter.org/, https://colab.google/) - inclusiv pentru activități de programare colaborativă;
- Raptor (https://raptor.martincarlisle.com/) - pentru reprezentare sub formă de schemă logică;
- MySQL Workbench (https://dev.mysql.com/workbench/).

**Interpretoare Python**

- CPython (www.python.org);
- PyPy (https://www.pypy.org/download.html).

Pentru **sprijinul activităților didactice de predare-învățare-evaluare** pot fi utilizate lecții interactive, tutoriale specifice, platforme de generare a testelor, ca cele recomandate mai jos.

Pentru generarea testelor:

- Kahoot! (https://kahoot.com), BookWidGets (https://www.bookwidgets.com/), Wayground (https://wayground.com/), Genially (https://genially.com/), WordWall (https://wordwall.net), Edcafe (https://www.edcafe.ai/).

Pentru obținerea unor mijloace de învățământ:

- Canva Education (https://www.canva.com/education) pentru creare de materiale vizuale, prezentări, postere și fișe de lucru; șabloane educaționale gratuite.
- MagicSchool (https://www.magicschool.ai/) platformă pentru crearea și distribuirea de resurse, lecții interactive și instrumente de evaluare.
- Livresq (https://livresq.com/ro/), bibliotecă de resurse educaționale interactive, platformă pentru crearea unor astfel de resurse;
- NEXTLAB.TECH (robo.nextlab.tech), un instrument modern de învățare prin practică, gamificare, inteligență artificială și proiecte interactive;
- Wolfram ALPHA (https://www.wolframalpha.com/) platformă de inteligență computațională care facilitează învățarea diverselor discipline.

Pentru învățare prin explorare și vizualizare, se recomandă utilizarea de diagrame și reprezentări grafice, simulatoare, instrumente interactive disponibile online, precum:

- HTML Maze (https://www.htmlmaze.online/maze) - instrument educațional pentru vizualizarea unor algoritmi, demonstrații pas cu pas etc.;
- Brainbashers (https://www.brainbashers.com/queens.asp) - instrument educațional pentru vizualizarea unor algoritmi, demonstrații pas cu pas etc.;
- Data Structure Visualizations (https://www.cs.usfca.edu/~galles/visualization/Algorithms.html) - o colecție interactivă de vizualizări pentru structuri de date și algoritmi
- Visualgo.net (https://visualgo.net/) - instrument educațional pentru vizualizarea algoritmilor, demonstrații pas cu pas pentru BFS/DFS, Dijkstra etc.;
- Graph Visualizer / Graph Online (https://graphonline.ru/en/) - instrument online de desenare și simulare a grafurilor orientate/neorientate, ponderate;
- CS Academy (https://csacademy.com/app/graph_editor/) - instrument online de desenare și simulare a grafurilor orientate/neorientate, ponderate;
- Algorithm Visualizer (https://algorithm-visualizer.org/) - platformă open-source pentru vizualizarea algoritmilor în mai multe limbaje (Python, JavaScript, C++), care conține animații pentru BFS și DFS, dar și pentru sortare, Backtracking, grafuri ponderate etc.;
- Python Tutor (https://pythontutor.com) – permite rularea pas cu pas a codului Python, C++ cu vizualizarea memoriei și a fluxului de execuție;
- CS Field Guide (https://csfieldguide.org.nz/en/) – resurse vizuale și jocuri interactive pentru concepte precum compresia datelor, criptare, rețele și algoritmi;
- Draw.io (https://app.diagrams.net) - pentru a desena entități, atribute, relații și a conecta elementele prin linii care se pot salva local sau în Google Drive;
- Lucidchart (https://lucid.app/) - platformă online dedicată creării de diagrame, care oferă modele predefinite pentru reprezentarea entităților și relațiilor;
- Teachable Machine (https://teachablemachine.withgoogle.com/) pentru a antrena modele simple (clasificare de imagini, recunoaștere de sunete), conectând activitățile la domenii diverse (biologie, arte, științe sociale);
- Machine Learning for Kids (https://machinelearningforkids.co.uk) pentru a antrena modele simple (clasificare de imagini, recunoaștere de sunete), conectând activitățile la domenii diverse (biologie, arte, științe sociale).

## GRUP DE LUCRU

| Nume și prenume | Grad didactic/Titlu științific | Instituție de apartenență, localitate, județ |
|---|---|---|
| CRĂCIUNESCU Georgeta Antonia Rodica | consilier | Ministerul Educației și Cercetării |
| ACIOBĂNIȚEI Iulian | conferențiar universitar, doctor | Academia Tehnică Militară „Ferdinand I”, București |
| ANTON Cristina Elena | profesor, gradul didactic I | Colegiul Național „Gh. M. Murgoci”, Brăila, județul Brăila |
| BOCA Alina Gabriela | profesor, gradul didactic I | Colegiul Național de Informatică „Tudor Vianu”, București |
| BORZA Diana-Laura | lector universitar, doctor | Universitatea Babeș Bolyai, Cluj-Napoca, județul Cluj |
| CÂRSTEA Claudia | conferențiar universitar, doctor | Academia Forțelor Aeriene „Henri Coandă”, Brașov, județul Brașov |
| CONTRAȘ Diana | profesor, gradul didactic I | Colegiul Național „Gheorghe Șincai”, Baia Mare, județul Maramureș |
| DIOSAN Laura-Silvia | profesor universitar, doctor | Universitatea Babeș-Bolyai, Cluj-Napoca, județul Cluj |
| DRAGOMIR-CONSTANTIN Florentina-Loredana | conferențiar universitar, doctor | Universitatea Națională de Apărare Carol I, București |
| DUMITRAN Adrian-Marius | lector universitar, doctor | Universitatea București, Facultatea de Matematică și Informatică |
| FLOREA Andrei | profesor, gradul didactic I | Colegiul Național „Ion Luca Caragiale”, București |
| IFTENE Adrian | profesor universitar, doctor | Universitatea „Alexandru Ioan Cuza”, Facultatea de Informatică, Iași, județul Iași |
| IORDAICHE Eugenia-Cristiana | profesor, gradul didactic I | Liceul Teoretic „Grigore Moisil”, Timișoara, județul Timiș |
| MARCU ALEXANDRA | profesor, gradul didactic I | Colegiul Național Militar „Tudor Vladimirescu”, Craiova, Dolj |
| MAIER Mariana-Ioana | lector universitar, doctor | Universitatea Babeș Bolyai, Cluj-Napoca, județul Cluj |
| MANZ Victor | profesor, gradul didactic I | Colegiul Național de Informatică „Tudor Vianu”, București |
| MARIUC Florin Constantin | profesor, gradul didactic I | Colegiul Național Militar „Ștefan cel Mare”, Câmpulung Moldovenesc, județul Suceava |
| MĂGUREANU Livia | cadru didactic asociat | Universitatea București, Facultatea de Matematică și Informatică |
| MODRIȘAN Adrian | profesor, gradul didactic I | Colegiul Național „Andrei Șaguna”, Brașov, județul Brașov |
| NAN Mihai | șef lucrări, doctor | Universitatea Națională de Științe și Tehnologie Politehnica București, Facultatea de Automatică și Calculatoare |

| Nume și prenume | Grad didactic/Titlu științific | Instituție de apartenență, localitate, județ |
|---|---|---|
| NIȚU Alina | profesor, gradul didactic I | Liceul Teoretic „Ovidius”, Constanța, județul Constanța |
| NURLA Aidan | profesor, gradul didactic I | Colegiul Naţional Militar „Alexandru Ioan Cuza”, Constanţa, județul Constanța |
| OANCEA Romana | conferențiar universitar, doctor | Academia Forţelor Terestre „Nicolae Bălcescu”, Sibiu, județul Sibiu |
| PETRESCU Manuela-Andreea | lector universitar, doctor | Universitatea Babeș-Bolyai, Cluj-Napoca, județul Cluj |
| PINTEA Adrian-Doru | profesor, gradul didactic I | Colegiul Național „Andrei Mureșanu”, Dej, județul Cluj |
| PINTEA Rodica | profesor, gradul didactic I | Liceul Teoretic „Radu Vlădescu”, Pătârlagele, județul Buzău |
| POP Florin | profesor universitar, doctor | Universitatea Națională de Științe și Tehnologie Politehnica București, Facultatea de Automatică și Calculatoare |
| POSTOLACHE Florin | lector universitar, doctor | Academia Navală „Mircea cel Bătrân” Constanța, Facultatea de Inginerie Marină, județul Constanța |
| SĂCUIU Silviu Eugen | profesor, gradul didactic I | Colegiul Național „Mihai Viteazul”, București |
| SPĂTARU Adrian | lector universitar, doctor | Universitatea de Vest, Timișoara, Facultatea de Matematică și Informatică, județul Timiș |
| STANCIU Diana | profesor, gradul didactic I | Colegiul Naţional Militar „Dimitrie Cantemir”, Breaza, județul Prahova |
| TEGLAȘ Maria | profesor, gradul didactic II | Colegiul Naţional Militar „Mihai Viteazul”, Alba Iulia, județul Alba |
| TÎMPLARU Roxana-Gabriela | profesor, gradul didactic I | Colegiul „Ștefan Obobleja”, Craiova, județul Dolj |
| UNGUREANU Florentina | profesor, gradul didactic I | Colegiul Național de Informatică, Piatra Neamț, județul Neamț |
| VINȚ Corina Elena | profesor, gradul didactic I | Colegiul Național de Informatică „Tudor Vianu”, București |

## COORDONATORI/RESPONSABILI/CONSULTANȚI ȘTIINȚIFICI

| Nume și prenume | Grad didactic/Titlu științific | Instituție de apartenență, localitate, județ |
|---|---|---|
| PENEA Ștefania | consilier | Centrul Național pentru Curriculum și Evaluare |
| ȚOCA Livia Demetra | consilier | Centrul Național pentru Curriculum și Evaluare |
| ȚĂPUS Nicolae | profesor universitar, doctor | Academia Română |
| BORIGA Radu Eugen | conferențiar universitar, doctor | Universitatea București, Facultatea de Matematică și Informatică |