



MONITORUL OFICIAL

AL

ROMÂNIEI

Anul 194 (XXXVIII) — Nr. 591 bis

PARTEA I
LEGI, DECRETE, HOTĂRĂRI ȘI ALTE ACTE

Luni, 20 iulie 2026

SUMAR

Pagina

Anexele nr. 1-11 la Ordinul ministrului educației și cercetării nr. 4.370/2026 privind aprobarea programelor școlare asociate disciplinei Informatică prevăzute în planurile- cadru pentru învățământul liceal	3-245
---	-------

ACTE ALE ORGANELOR DE SPECIALITATE ALE ADMINISTRAȚIEI PUBLICE CENTRALE

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII

ORDIN

privind aprobarea programelor școlare asociate disciplinei Informatică prevăzute în planurile-cadru pentru învățământul liceal*)

În conformitate cu prevederile art. 88 alin. (2) și (10) din Legea învățământului preuniversitar nr. 198/2023, cu modificările și completările ulterioare,

având în vedere prevederile art. 4 din Hotărârea Guvernului nr. 251/2025 privind organizarea și funcționarea Centrului Național pentru Curriculum și Evaluare,

având în vedere Ordinul ministrului educației nr. 6.072/2023 privind aprobarea unor măsuri tranzitorii aplicabile la nivelul sistemului național de învățământ preuniversitar și superior, cu modificările ulterioare,

în baza Ordinului ministrului educației naționale nr. 3.593/2014 pentru aprobarea Metodologiei privind elaborarea și aprobarea curriculumului școlar — planuri-cadru de învățământ și programe școlare,

ținând cont de Ordinul ministrului educației și cercetării nr. 4.350/2025 privind aprobarea planurilor-cadru pentru învățământul liceal cu frecvență zi, cu modificările ulterioare, și de Ordinul ministrului educației și cercetării nr. 6.873/2025 privind aprobarea planurilor-cadru pentru învățământul liceal special cu frecvență zi, învățământul liceal cu frecvență zi, pentru clasele cu predare a unei limbi moderne în regim intensiv/bilingv, învățământul liceal cu frecvență zi, pentru clasele cu predare a disciplinei informatică în regim intensiv, și învățământul liceal cu frecvență seral/redușă,

având în vedere Referatul de aprobare nr. 1.539 din 8.07.2026,

în temeiul art. 13 alin. (3) din Hotărârea Guvernului nr. 731/2024 privind organizarea și funcționarea Ministerului Educației și Cercetării, cu modificările și completările ulterioare,

ministrul educației și cercetării emite prezentul ordin.

Art. 1. — Se aprobă Centralizatorul programelor școlare asociate disciplinei Informatică, conform planurilor-cadru pentru învățământul liceal, prevăzut în anexa nr. 1.

Art. 2. — Se aprobă programele școlare asociate disciplinei Informatică, prevăzute în planurile-cadru pentru învățământul liceal, prevăzute în anexele nr. 2-11.

Art. 3. — Programele școlare prevăzute în anexele nr. 2-11 se aplică progresiv începând cu clasa a IX-a a anului școlar 2026-2027.

Art. 4. — Programele școlare asociate fiecărui an de studiu sunt valabile pe o perioadă de 8 ani, începând cu anii școlari 2026-2027 pentru clasa a IX-a, 2027-2028 pentru clasa a X-a,

2028-2029 pentru clasa a XI-a, respectiv 2029-2030 pentru clasa a XII-a.

Art. 5. — Anexele nr. 1-11 fac parte integrantă din prezentul ordin.

Art. 6. — Direcția generală echitate și performanță în învățământul preuniversitar, Direcția generală minorități și desegregare, Centrul Național pentru Curriculum și Evaluare, inspectoratele școlare și unitățile de învățământ duc la îndeplinire prevederile prezentului ordin.

Art. 7. — Prezentul ordin se publică în Monitorul Oficial al României, Partea I.

Ministrul educației și cercetării,

Mihai Dimian

București, 9 iulie 2026.

Nr. 4.370.

*) Ordinul nr. 4.370/2026 a fost publicat în Monitorul Oficial al României, Partea I, nr. 591 din 20 iulie 2026 și este reprodus și în acest număr bis.

Anexa nr. 1

**CENTRALIZATORUL PROGRAMELOR ȘCOLARE ASOCIATE DISCIPLINEI INFORMATICĂ,
CONFORM PLANURILOR-CADRU PENTRU ÎNVĂȚĂMÂNTUL LICEAL**

Numărul anexei la ordin	Disciplină	Tip programă
2.	Informatică	Programa școlară pentru disciplina Informatică, clasa a X-a, curriculum de specialitate (CS), filiera teoretică, profilul real, specializarea matematică-informatică
3.	Informatică	Programa școlară pentru disciplina Informatică, clasa a XI-a, curriculum de specialitate (CS), filiera teoretică, profilul real, specializarea matematică-informatică
4.	Informatică	Programa școlară pentru disciplina Informatică, clasa a XII-a, curriculum de specialitate (CS); filiera teoretică, profilul real, specializarea matematică-informatică; filiera vocațională, profilul militar, specializarea matematică-informatică militară
5.	Informatică	Programa școlară pentru disciplina Informatică, clasa a X-a, curriculum de specialitate (CS), filiera vocațională, profilul militar, specializarea matematică-informatică militară
6.	Informatică	Programa școlară pentru disciplina Informatică, clasa a XI-a, curriculum de specialitate (CS), filiera vocațională, profilul militar, specializarea matematică-informatică militară
7.	Informatică	Programa școlară pentru disciplina Informatică, clasa a X-a, curriculum de specialitate (CS), filiera teoretică, profilul real, specializarea științe ale naturii
8.	Informatică	Programa școlară pentru disciplina Informatică, clasa a IX-a, curriculum de specialitate (CS), filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei informatice în regim intensiv
9.	Informatică	Programa școlară pentru disciplina Informatică, clasa a X-a, curriculum de specialitate (CS), filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei informatice în regim intensiv
10.	Informatică	Programa școlară pentru disciplina Informatică, clasa a XI-a, curriculum de specialitate (CS), filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei informatice în regim intensiv
11.	Informatică	Programa școlară pentru disciplina Informatică, clasa a XII-a, curriculum de specialitate (CS), filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei informatice în regim intensiv

Anexa nr. 2

Programa școlară
pentru disciplina

Informatică

Clasa a X-a

Curriculum de specialitate (CS)

Filiera teoretică, profilul real, specializarea matematică-informatică

Învățământ liceal

- 2026 –

Anexa nr. 2**Programa școlară pentru disciplina Informatică, clasa a X-a, curriculum de specialitate (CS)**

- *Filiera teoretică, profilul real, specializarea matematică-informatică*

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 matematică-informatică, alocarea orară este:

- clasa a IX-a – 2 ore/săptămână, dintre care o oră pentru studiu teoretic și o oră pentru activități practice;
- clasa a X-a – 2 ore/săptămână, dintre care o oră pentru studiu teoretic și o oră pentru activități practice;
- clasa a XI-a – 4 ore/săptămână, dintre care două ore pentru studiu teoretic și două ore pentru activități practice;
- clasa a XII-a – 3 ore/săptămână, dintre care o oră pentru studiu teoretic și două ore pentru 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** (CDEOS), prin opționale care pot fi de aprofundare, noi discipline/module de pregătire sau opționale integrate prin care elevii pot primi sprijin pentru remediare, 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 limbajelor de programare Python și 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 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. 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 experiența 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 - neliniare, liniare, asociative, mixte, pentru a structura și accesa date în vederea prelucrării acestora	
- <i>definirea conceptului de mulțime în contextul determinării drepturilor distincte de acces care sunt atribuite unui grup de utilizatori cu roluri diferite, unde fiecare drept apare o singură dată, chiar dacă e atribuit mai multor persoane, iar ordinea înregistrării acestora nu contează</i> - <i>enumerarea operațiilor de bază asupra mulțimilor (reuniune, intersecție, diferență, egalitate, incluziune, testare apartenență), în contextul în care se cunoaște lista obiectelor transportate cu două rucsacuri și se cere să fie identificate obiectele comune, cele conținute de un singur rucsac etc.</i> - <i>precizarea caracteristicilor unei mulțimi în contextul gestionării utilizatorilor înregistrați pe platforme diferite pentru a evita dublurile</i> - <i>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</i> - <i>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</i> - <i>definirea conceptului de structură asociativă (dicționar) cu evidențierea faptului că se rețin perechi cheie-valoare și se permite accesul rapid la date prin intermediul cheilor, de exemplu, într-o aplicație de traducere, în care fiecare cuvânt este o „cheie”, iar traducerea sa este „valoarea” sau în contextul asocierii dicționarului cu agenda telefonică în care fiecărui număr de telefon (cheie) îi corespunde un nume (valoare)</i> - <i>enumerarea cheilor și a valorilor asociate dintr-un dicționar pentru un formular online, în care fiecare câmp completat (de exemplu, „nume”, „vârstă”, „email”) este asociat cu valorile corespunzătoare</i> - <i>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;</i> - <i>identificarea caracteristicilor unui model mixt, de tipul dicționar de liste și a modului de organizare a datelor în contextul evidenței școlare a claselor, în care fiecare clasă are o listă de elevi</i> - <i>enumerarea metodelor de acces și parcurgere a datelor într-o listă de dicționare în contextul gestiunii unor comenzi, fiecare comandă fiind reprezentată printr-un dicționar (prin asocierea produs-cantitate) pentru a genera facturile corespunzătoare</i>	
CS 1.2. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a fi putea utiliza în prelucrarea listelor	
- <i>recunoașterea caracteristicilor metodei căutării binare cu specificarea condiției necesare, în contextul localizării rapide a unui contact într-o agendă telefonică ordonată alfabetic</i> - <i>enumerarea etapelor metodei căutării binare în contextul localizării unui titlu într-o listă ordonată de cărți dintr-o bibliotecă digitală</i> - <i>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</i> - <i>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</i>	
CS 1.3. Identifică specificul, caracteristicile și etapele unor strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor	
- <i>definirea caracteristicilor metodei Divide et impera în contextul unei discuții ghidate în care sunt reliefate avantajele metodei prin analogie cu fragmentarea unei activități complexe în subactivități realizate de echipe diferite, urmate de reunirea rezultatelor</i> - <i>enumerarea pașilor specifici metodei Divide et impera (divizarea problemei, rezolvarea subproblemelor, combinarea soluțiilor) în contextul aplicării acestei strategii la algoritmul de determinare a valorii maxime dintr-un șir de numere memorate</i> - <i>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</i>	

Clasa a X-a	
-	<i>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</i>
CS 1.4. Identifică principalele elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele simple - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice	
-	<i>recunoașterea sintaxei corecte de inițializare a unei mulțimi (set) în contextul în care sunt adunate obiecte diferite (de exemplu, tipuri de fructe) într-o colecție în care elementele sunt unice, cu evidențierea modului în care utilizarea acoladelor și a elementelor separate prin virgule permite crearea unei mulțimi fără dubluri și fără ordine prestabilită</i>
-	<i>identificarea operatorului pentru verificarea apartenenței unui element la o mulțime în Python în contextul scrierii unei secvențe de instrucțiuni prin care se verifică dacă o persoană este deja înscrisă pe lista participanților la un eveniment, fără a fi necesară parcurgerea întregii liste</i>
-	<i>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.)</i>
-	<i>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</i>
-	<i>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</i>
-	<i>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</i>
-	<i>recunoașterea notărilor specifice cu acolade și a separatorului între cheie și valoare în cazul dicționarelor, prin analogie cu organizarea informațiilor într-o listă de contacte: numele persoanei urmat de simbolul „:”, apoi de numărul de telefon, elementele fiind grupate între acolade</i>
-	<i>enumerarea operațiilor de bază asupra dicționarelor (creare, adăugare, ștergere, actualizare) prin analogie cu actualizarea listei de contacte din agenda telefonică: crearea unei liste de contacte, adăugarea unui contact nou, ștergerea unui contact existent, modificarea numărului de telefon al unei persoane</i>
-	<i>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</i>
CS 1.5. Identifică elementele de sintaxă și componentele din definiția și apelul subprogramelor recursive, pentru a le utiliza în implementarea algoritmilor în limbaj de programare	
-	<i>enumerarea caracteristicilor principale ale unui subprogram recursiv, în contextul elaborării unui astfel de subprogram pentru calculul lui $n!$ definit prin relația recurentă cunoscută</i>
-	<i>recunoașterea termenilor „recursivitate directă” și „recursivitate indirectă” prin asocierea lor cu definițiile corespunzătoare</i>
-	<i>recunoașterea sintaxei corecte pentru definirea unui subprogram recursiv în Python, în contextul calculării sumei cifrelor unui număr</i>
-	<i>enumerarea condițiilor pe care trebuie să le îndeplinească un subprogram recursiv pentru a funcționa corect, menționând necesitatea unui caz de oprire, a unei transformări care reduce progresiv problema și a unui apel recursiv care aplică aceeași procedură până la atingerea condiției de oprire, în contextul calculării factorialului unui număr</i>

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 - neliniare, liniare, asociative, mixte, pentru a le gestiona eficient	
-	<i>explicarea caracteristicilor mulțimilor și listelor, în contextul organizării și accesării elementelor acestora în situații cotidiene, de exemplu, o listă de persoane care așteaptă la rând și determinarea clienților distincți ai unei companii de telefonie, pe baza unei mulțimi de abonamente (în condițiile în care un client poate avea mai multe astfel de abonamente)</i>
-	<i>explicarea modului în care mulțimile sunt utilizate pentru clasificarea pe categorii a produselor într-un magazin online, evidențiind cum apartenența la o categorie permite organizarea și filtrarea eficientă a articolelor</i>
-	<i>explicarea modului în care este prelucrat un șir de caractere, în contextul funcționării unui sistem de verificare a complexității parolilor, care analizează fiecare simbol pentru a identifica litere mari, litere mici, cifre și caractere speciale</i>

Clasa a X-a	
- 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 - explicarea rolului cheilor unice și a valorilor asociate în dicționare, în contextul serviciului de evidență populației, unde nu pot exista două CNP-uri identice (chei unice), iar modificarea numelui unei persoane actualizează valoarea corespunzătoare cheii respective - explicarea modului în care un dicționar poate fi folosit pentru a organiza toate informațiile unei comenzi online, fiecare cheie din dicționar reprezentând un tip de informație (de exemplu, „client”, „adresă”, „produse”), iar valoarea asociată acelei chei conținând detaliile respective, astfel încât toate datele comenzii să fie grupate logic și să fie ușor de găsit - 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; - explicarea modului în care un dicționar este folosit pentru a structura datele într-un sistem de gestiune a proiectelor, fiecare cheie reprezentând numele unei echipe, iar valoarea asociată fiind lista sarcinilor acelei echipe, ceea ce permite vizualizarea clară și separată a responsabilităților fiecărei echipe - explicarea modului în care structura unui orar poate fi reprezentată printr-un dicționar, în care fiecare cheie corespunde unei zile, iar valoarea asociată este lista orelor din acea zi, ceea ce permite accesul rapid la programul complet al unei anumite zile	
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 - 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, identificând eventualele suprapuneri) - 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 Divide et impera, Greedy pentru rezolvarea problemelor	
- explicarea etapelor metodei Divide et impera (descompunere, rezolvare, combinare), folosind ca exemplu calcularea sumei unor numere dintr-o listă, prin împărțirea repetată a acesteia și a fiecărei liste obținute pe parcurs în jumătăți (până când se ajunge la liste formate dintr-un singur număr, pentru care se cunoaște rezultatul) și determinarea rezultatului pentru o astfel de listă, prin adunarea sumelor corespunzătoare celor două jumătăți, după obținerea acestora - explicarea modului în care se aplică metoda Divide et impera, descriind clar cele trei etape ale sale — descompunerea problemei în subprobleme mai mici, rezolvarea fiecărei subprobleme, apoi combinarea rezultatelor în exemple reale în care această abordare este folosită, precum procesarea imaginilor sau navigația GPS - 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 - 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 $3+3$) - 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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice	
- explicarea modalității de parcurgere a elementelor unei mulțimi în Python prin analogie cu procesul de verificare a fiecărui obiect dintr-o cutie care conține piese unice, fără a ține cont de ordinea lor, evidențiind modul în care o instrucțiune repetitivă permite accesarea fiecărui element al mulțimii - explicarea efectului aplicării metodelor (<code>add()</code>, <code>remove()</code>, <code>union()</code>, <code>intersection()</code>) și operatorilor (<code> </code>, <code>&</code>, <code>-</code>) asupra a două mulțimi în Python, în contextul ilustrării în limbaj de programare a unor exemple inspirate din viața cotidiană (de exemplu, compararea listelor de prieteni din două rețele sociale, respectiv a prietenilor existenți în oricare rețea, a prietenilor comuni, a prietenilor prezenți doar în una dintre rețele)	

Clasa a X-a	
- 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()) - explicarea efectului aplicării unor metode specifice dicționarilor prin asocierea acestora cu gestionarea informațiilor din agenda telefonică, de exemplu, verificarea faptului că un nume se află în agendă (get()), obținerea listei care conține doar numele din agendă (keys()), obținerea listei care conține doar numerele de telefon din agendă (values()), obținerea listei complete nume-număr din agendă (items()) - explicarea rezultatelor obținute în urma iterării asupra unui dicționar, cu exemplificare pe agenda telefonică, din care, în urma iterării, se poate obține doar lista de nume, doar lista de numere de telefon sau lista de perechi nume-număr de telefon, în funcție de metoda de parcurgere utilizată - 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	
CS 2.5. Explică mecanismul de executare a subprogramelor recursive, pentru a le utiliza în implementarea algoritmilor în limbaj de programare	
- explicarea modului de executare a unui subprogram recursiv, descriind cum fiecare apel este plasat pe stivă și rămâne în așteptare până la rezolvarea apelului următor, folosind ca exemplu o serie de cutii puse una în alta: întâi se deschide cutia cea mai mare, apoi cutia aflată în interiorul ei și tot așa până la cutia cea mai mică; abia după ce se ajunge la ultima cutie și se preia obiectul plasat în aceasta, cutiile sunt închise în ordinea inversă deschiderii, exact cum apelurile recursive sunt rezolvate în ordine inversă plasării lor pe stivă - explicarea modului în care un algoritm iterativ poate fi transformat într-o variantă recursivă echivalentă, folosind ca exemplu calcularea sumei primelor n numere naturale: varianta iterativă adună pe rând numerele de la 1 la n într-un ciclu, iar varianta recursivă calculează suma exprimând-o ca valoarea curentă plus suma numerelor rămase - explicarea modului în care apelul recursiv al unui subprogram conduce la reluarea instrucțiunilor acestuia, într-un alt context, mai restrâns, pentru rezolvarea unor probleme matematice precum calcularea factorialului, determinarea celui de-al n-lea termen al unui șir definit recursiv sau calculul celui mai mare divizor comun prin scăderi repetate, evidențiind necesitatea unui caz de oprire și a faptului că funcția se apelează repetat	

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 - neliniare, liniare, asociative, mixte, pentru a rezolva probleme de gestionare a datelor	
- aplicarea operațiilor de reuniune și intersecție asupra a două mulțimi de clienți din două magazine online, pentru a stabili atât lista tuturor clienților care pot primi oferte promoționale, cât și clienții comuni celor două magazine - utilizarea mulțimilor pentru generarea elementelor unice (de exemplu, coduri de acces fără repetiție), verificarea dublurilor într-o listă de elemente (de exemplu, produse identice într-un inventar), verificarea faptului că o colecție conține toate valorile dintr-o secvență de elemente (de exemplu, verificarea faptului că toate zilele unei săptămâni au fost completate într-un raport) - utilizarea comparării lexicografice pentru ordonarea titlurilor de cărți într-o bibliotecă digitală, astfel încât acestea să fie afișate alfabetic într-o listă - 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) - 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 - utilizarea organizării datelor sub formă de dicționar pentru realizarea unui glosar online care asociază niște termeni din tehnologia informației (chei) cu explicațiile lor (valori), facilitând învățarea rapidă a vocabularului de specialitate - utilizarea dicționarilor pentru organizarea perechilor cheie-valoare și obținerea unor rezultate, în contextul parcurgerii listei de produse dintr-un magazin unde fiecare produs (cheia) are asociat un preț (valoarea) și obținerea de rezultate precum valoarea totală a stocului, prețul minim/maxim, media prețurilor, produsele cu prețul mai mare/mai mic decât o valoare dată etc.	

Clasa a X-a	
- accesarea unui element dintr-un dicționar utilizat într-o aplicație de comerț online, în care codul unic al produsului este folosit drept cheie, iar valorile asociate sunt informațiile despre produs (denumire și preț), pentru a afișa rapid detaliile corespunzătoare codului introdus de utilizator - 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 - 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 - 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 - 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 Divide et impera, 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 - 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) - aplicarea metodei de sortare prin interclasare pentru obținerea unei liste ordonate pornind de la o listă neordonată formată din 7 numere date - 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 - 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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice	
- utilizarea unor operatori (, &, -) și metode (add(), remove(), union(), intersection()) ale clasei set în contexte practice, de exemplu, pentru gestionarea participării elevilor la activități extrașcolare, cu evidențierea modului în care aceste operații pot simplifica prelucrarea și compararea datelor - utilizarea unor metode adecvate din clasa set în implementarea unor programe pentru verificarea faptului că o listă este formată din elemente distincte (de exemplu, coduri unice de participare), eliminarea elementelor duplicate dintr-o listă (de exemplu, coduri introduse de mai multe ori), determinarea elementelor comune între două colecții (de exemplu, elevii înscriși la două cercuri diferite) - 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()) - 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()) - aplicarea metodelor uzuale ale clasei dict în contexte practice, de exemplu, pentru gestionarea produselor dintr-un magazin: actualizarea stocului de produse (update()), ștergerea unui articol din magazin (pop()), ștergerea ultimului produs adăugat (popitem()), lichidare de stoc (clear()), crearea unei copii de siguranță a stocului înainte de modificări majore (copy()), crearea unui nou stoc pornind de la un stoc existent (fromkeys()) în care vor fi toate produsele inițiale (cheile) dar prețurile aferente (valorile) au valoarea None, verificarea existenței unui produs în magazin cu scopul obținerii prețului acestuia și evitarea erorilor în cazul în care produsul lipsește (get()), adăugarea automată a unui produs în magazin dacă nu există deja (setdefault()), obținerea listelor cu denumirile produselor pentru inventar (keys()), prețurile din magazin (values()) sau perechile produs-preț (items()) pentru afișare pe etichete	

Clasa a X-a	
- utilizarea de dicționare imbricate, cu exemplificare pe gestionarea stocului unui magazin (reprezentat sub formă de dicționar) în care fiecare produs (cheia) are asociat un alt dicționar cu detalii ca preț, cantitate etc. - 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	
CS 3.5. Utilizează elementele de sintaxă și componentele din definiția și apelul subprogramelor recursive, în implementarea algoritmilor în limbaj de programare	
- aplicarea elementelor de sintaxă pentru implementarea unui subprogram recursiv, pe baza unei relații recurente date, care determină factorialul unui număr, de exemplu, pentru stabilirea numărului de moduri diferite în care pot fi așezate 4 persoane pe 4 scaune într-o sală - aplicarea recursivității pentru determinarea unui termen din șirul lui Fibonacci, prin urmărirea apelurilor unui subprogram recursiv în Python - aplicarea recursivității pentru calcularea sumei elementelor unei liste, pe baza unui subprogram recursiv dat în limbajul Python, care adună elementul curent al listei cu rezultatul obținut prin apelarea sa asupra restului listei - implementarea unui subprogram recursiv pornind de la o relație de recurență dată, de exemplu, pentru calculul combinațiilor de câte k elemente dintre cele n date, pe baza formulei $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ cu respectarea condițiilor matematice	

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 - neliniare, liniare, asociative, mixte, pentru a alege structura adecvată în rezolvarea unor probleme concrete	
- analizarea avantajelor și limitărilor utilizării mulțimilor în comparație cu listele, prin raportare la situații practice, precum verificarea rapidă a înscrierii unui elev în catalogul clasei, cu evidențierea faptului că verificarea apartenenței unui element este mai eficientă din punctul de vedere al numărului de operații efectuate, utilizând mulțimi comparativ cu liste, dar acestea nu păstrează ordinea și nu acceptă elemente duplicat - 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 - 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) - analizarea avantajelor utilizării unui dicționar într-o aplicație financiară, în care codul fiscal al fiecărui client este utilizat drept cheie pentru accesarea directă a datelor asociate, astfel încât informațiile despre un client sunt obținute imediat, fără a fi nevoie de verificarea succesivă a tuturor celorlalți clienți din evidență - analizarea comportamentului datelor organizate în dicționare la actualizarea valorilor pentru chei existente versus chei noi, în contextul modificării prețului unui produs existent (cheie existentă) sau la adăugarea unui produs nou (cheie nouă) într-un magazin - analizarea modului în care este utilizat un dicționar într-un sistem de evidență al unei biblioteci, în care fiecare carte este identificată printr-o cheie (de exemplu, codul cărții), iar valoarea asociată indică plasarea acesteia, pentru a verifica dacă fiecărei chei îi corespunde o plasare, evitând astfel pierderea informației despre plasarea cărților în bibliotecă - 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 - 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 - 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 - 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	

Clasa a X-a	
<i>finală, cu toți clienții, ordonată cronologic după momentul sosirii, prin metoda interclasării, respectiv prin concatenarea celor două liste și ordonarea listei obținute</i> - <i>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</i>	
CS 4.3. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor	
- <i>analizarea aplicării metodei Divide et impera printr-un exemplu concret, în care o listă de scoruri ale jucătorilor este împărțită repetat în jumătate, până când se ajunge la liste formate dintr-un singur scor, fiecare sublistă este prelucrată separat prin aceeași regulă (de exemplu, determinarea scorului maxim/minim din sublistă), iar rezultatele obținute sunt apoi comparate pentru a stabili scorul maxim din lista inițială, evidențiind explicit etapele de împărțire, rezolvare și combinare</i> - <i>analizarea procesului de împărțire succesivă a unei probleme prin metoda Divide et impera, printr-un exemplu precum calculul sumei elementelor unei liste</i> - <i>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</i> - <i>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</i> - <i>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</i>	
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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice	
- <i>analizarea mesajelor de eroare apărute în Python la includerea într-o mulțime a unor elemente mutabile (de exemplu, liste sau dicționare), prin asocierea cu situația în care într-o colecție de obiecte unice se încearcă adăugarea unui obiect care își poate schimba forma (de exemplu, o trusă de geometrie ce poate conține doar reprezentări ale unor corpuri geometrice care sunt rigide, deci nu se pot modifica, comparativ cu cazul în care acestea ar fi realizate din plastilină, astfel încât să poată fi remodelate)</i> - <i>analizarea comportamentului unor programe care realizează operații de bază cu mulțimi (de exemplu, verificări rapide ale apartenenței, identificarea dublurilor într-o listă de înscrieri), utilizând clasa set, respectiv list, pentru alegerea unei variante eficiente din punctul de vedere al timpului de executare</i> - <i>analizarea avantajelor utilizării unei liste de șiruri de caractere, comparativ cu un șir compact, pentru afișarea în ordine alfabetică a unor cuvinte</i> - <i>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</i> - <i>analizarea avantajelor utilizării dicționarelor în contextul comparării stocurilor a două magazine pentru a identifica produse disponibile în ambele magazine (chei comune), produse care există doar într-un magazin și nu în celălalt (diferențe)</i> - <i>analizarea modului în care pot fi combinate dicționarele, în contextul unificării stocurilor din mai multe magazine, pentru adăugarea sau actualizarea datelor produselor din magazinul principal (update()), unificarea stocurilor pentru generarea unui catalog de produse (operatorii “ ”, “**”), unificarea manuală și verificarea cheilor duplicate, filtrarea produselor cu preț mai mare decât o valoare dată în momentul unificării</i> - <i>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</i>	
CS 4.5. Analizează aplicabilitatea, avantajele și dezavantajele utilizării subprogramelor recursive, în implementarea algoritmilor în limbaj de programare	
- <i>analizarea consumului de timp și memorie al unui subprogram recursiv în Python care rezolvă o problemă matematică, precum determinarea sumei primelor n numere naturale, prin observarea numărului de apeluri recursive efectuate și a spațiului ocupat în stiva de executare pe măsură ce parametrul utilizat se modifică</i>	

Clasa a X-a
- compararea abordării recursive și a celei iterative în Python pentru rezolvarea unei probleme matematice precum calculul sumei primelor n numere naturale, analizând diferențele dintre cele două moduri de rezolvare în ceea ce privește numărul de pași efectuați, modul de repetare a operațiilor și utilizarea memoriei în funcție de valoarea lui n - compararea a doi algoritmi pentru determinarea unui termen al șirului lui Fibonacci, unul iterativ, iar celălalt recursiv, pe baza desfășurării pas cu pas a executării acestuia, urmărind ordinea apelurilor și revenirea din acestea, prin analogie cu verificarea succesivă a unor cutii puse una în alta, unde fiecare verificare nouă este memorată temporar până când se ajunge la ultima cutie și se revine treptat la prima, observând reluarea unor calcule

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 - neliniare, liniare, asociative, mixte și a modurilor de prelucrare a acestora pentru rezolvarea unor probleme informatice concrete
- argumentarea adecvării utilizării mulțimilor care gestionează date neordonate și fără elemente duplicate (de exemplu, administrarea codurilor unice ale produselor dintr-un magazin, evidența participanților la un eveniment) cu justificarea alegerii acestora pentru avantajele de unicatitate - evaluarea avantajelor utilizării mulțimilor în organizarea unui volum mare de informații despre clienți, în comparație cu alte forme de organizare precum listele, justificând alegerea mulțimilor prin capacitatea lor de a elimina automat aparițiile repetate și de a permite identificarea rapidă a elementelor unice, ceea ce reduce erorile și timpul de prelucrare a datelor - 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 - 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 - evaluarea corectitudinii utilizării unui dicționar digital într-o aplicație reală (de exemplu, un catalog electronic de produse), prin verificarea faptului că fiecare cheie este unică și că valoarea asociată conține informații complete și corecte, justificând că această structură previne confuziile și erorile de afișare în timpul utilizării - justificarea alegerii dicționarului pentru rezolvarea unor probleme din context cotidian (de exemplu, gestionarea unui catalog de elevi, a unei biblioteci, a unui concurs sportiv, a unei agenții de turism, organizarea informațiilor economice cum ar fi asocierea codurilor de factură cu datele corespunzătoare ale clienților), prin accesul rapid la date și prin claritatea relației dintre identificator și informația asociată - evaluarea avantajelor utilizării unei structuri de tip dicționar care asociază termeni cu definițiile lor (de exemplu, un glosar digital), prin aprecierea corectitudinii relației termen–definiție și justificarea modului în care această organizare sprijină regăsirea rapidă și fără ambiguități a informațiilor - evaluarea corectitudinii datelor într-un model conceptual mixt (de exemplu, listă de liste, listă de dicționare) 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) - argumentarea alegerii unui model mixt pentru un sistem de catalog electronic (de exemplu, listă de liste, listă de dicționare) 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 - 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 - 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 Divide et impera, Greedy pentru rezolvarea problemelor
- evaluarea oportunității aplicării metodei Divide et impera în contexte matematice (de exemplu, căutarea unui element într-un interval numeric prin înjumătățiri succesive), prin justificarea deciziei pe baza modului de împărțire a problemei în subprobleme independente, a reducerii numărului de pași și a eficienței soluției finale - evaluarea corectitudinii unui algoritm bazat pe metoda Divide et impera utilizat într-o problemă matematică (de exemplu, calculul rapid al unei puteri sau al maximului dintr-un șir de valori), prin justificarea validității soluției obținute pentru toate subproblemele și a modului în care acestea sunt combinate

Clasa a X-a	
- 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 - evaluarea obținerii soluției optime pentru probleme rezolvate cu metoda Greedy (de exemplu, organizarea unui program de întâlniri fără suprapuneri), prin justificarea corectitudinii condițiilor alese - 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, $\text{card}(A)=n$ și $\text{card}(B) \geq n$), 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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice	
- evaluarea corectitudinii și eficienței diferitelor soluții de eliminare a elementelor duplicate dintr-o listă, prin compararea implementărilor iterative cu cele bazate pe conversia listei într-o mulțime (set), pentru a evidenția avantajele utilizării mulțimilor în simplificarea și accelerarea procesului de filtrare - evaluarea eficienței a două soluții pentru aceeași problemă, una care integrează operații specifice clasei set pentru îmbunătățirea clarității și performanței și alta bazată pe liste (de exemplu, gestionarea datelor de înscriere la un eveniment, filtrarea elementelor unice dintr-un fișier) - evaluarea eficienței și clarității redactării a două soluții pentru aceeași problemă, care verifică dacă un cuvânt este anagramă a unui alt cuvânt - argumentarea alegerii unei metode din clasa str pentru o sarcină practică, de exemplu, căutarea unor vocale într-un text dat - evaluarea eficienței unei soluții bazate pe dicționar față de una bazată pe listă de liste (listă de perechi), în contextul căutării rapide a prețului unui produs într-un catalog - evaluarea unui cod care folosește inadecvat metodele clasei dict (de exemplu, încercarea de a accesa o cheie inexistentă fără metoda get(), utilizarea metodei pop() cu argument greșit, încercarea de a modifica o cheie etc.) - 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	
CS 5.5. Evaluează corectitudinea și eficiența programelor care utilizează subprograme recursive, pentru a rezolva probleme informatice	
- evaluarea eficienței unui subprogram recursiv în Python în raport cu o implementare iterativă echivalentă pentru o problemă matematică (de exemplu, calculul factorialului unui număr), prin justificarea diferențelor de claritate a soluției, număr de pași de executare și consum de memorie - evaluarea limitărilor în utilizarea unui subprogram recursiv în Python pentru rezolvarea unei probleme matematice (de exemplu, determinarea celui de al n-lea termen al șirului lui Fibonacci), evidențiind apeluri repetate pentru realizarea unor calcule identice - evaluarea deciziei de folosire a unui subprogram recursiv în Python pentru organizarea unui buget complex (de exemplu, calcularea soldului lunar pentru un buget structurat ierarhic, folosind liste de liste sau dicționare de dicționare), prin justificarea modului în care structura recursivă simplifică tratarea nivelurilor multiple de date față de o soluție iterativă	

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 - neliniare, liniare, asociative, mixte, pentru organizarea și prelucrarea eficientă a datelor, utilizând algoritmi de bază, în vederea rezolvării unor probleme	
- proiectarea unui algoritm de păstrare a evidențelor pentru o organizație educațională (de exemplu, o platformă de cursuri), pentru a identifica participanții înscriși, a evita înscrierile redundante, pe baza datelor utilizatorilor care sunt organizate în mulțimi - proiectarea unui algoritm pentru analizarea datelor unui magazin online, în care sunt definite explicit mulțimea tuturor clienților înregistrați și, pentru fiecare produs, mulțimea clienților care l-au achiziționat, iar relațiile dintre mulțimi sunt utilizate pentru recomandarea de perechi de produse care sunt achiziționate de obicei împreună - 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	

Clasa a X-a	
- 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 - proiectarea unui algoritm de actualizare automată a unui dicționar de cursuri valutare, în care cheia este codul monedei (de exemplu, EUR, USD), iar valoarea este cursul de schimb asociat în lei, algoritmul preluând periodic date noi și înlocuind valorile existente cu cele actualizate - proiectarea unui sistem de înlocuire automată a termenilor dintr-un text, folosind un dicționar în care cheia este cuvântul sau expresia căutată, iar valoarea este cuvântul sau expresia care o înlocuiește, algoritmul parcurgând textul și substituind fiecare apariție a cheilor cu valorile corespunzătoare - proiectarea unui algoritm care utilizează un model mixt (listă de liste sau dicționar 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 - proiectarea unui algoritm de filtrare a comenzilor dintr-o listă în care pentru fiecare înregistrare se păstrează suma plătită și numărul bonului, pentru a afișa doar comenzile care depășesc o sumă prestabilită	
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 - 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 - 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	
CS 6.3. Proiectează algoritmi cu aplicarea personalizată a strategiilor de tip Divide et impera, Greedy pentru rezolvarea problemelor	
- proiectarea unor algoritmi care utilizează metoda Divide et impera, pentru contexte reale (de exemplu, căutarea unui document într-o arhivă ordonată), prin definirea adecvată a datelor și a subproblemelor, astfel încât rezultatul să fie obținut împărțind problema inițială în subprobleme mai mici de același tip - proiectarea unor algoritmi pentru crearea unor fractali simpli (de exemplu, covorul lui Sierpinski), utilizând metoda Divide et impera - proiectarea unor soluții informatice care aplică personalizat metoda Divide et impera pentru problema turnurilor din Hanoi prin reprezentarea configurației tijelor folosind structuri de date adecvate și prin afișarea etapizată a mutărilor efectuate - 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 - 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 - neliniare, liniare, asociative, mixte, în vederea realizării unor soluții funcționale și performante	
- implementarea unor aplicații personalizate care utilizează clase Python specifice pentru gestionarea mulțimilor la matematică (de exemplu, un software educațional cu titlul „Mulțimi”) sau generarea unor numere distincte pentru jocuri de tip Bingo - implementarea unor aplicații personalizate pentru prelucrarea cifrelor unui număr utilizând adecvat clasa set din Python (de exemplu, determinarea cifrelor distincte ale unui număr, formarea unui număr maxim cu cifrele distincte ale unui număr dat, determinarea cifrelor comune ale unor numere date, determinarea cifrelor care nu apar în scrierea unui număr etc.) - 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’) - 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)	

Clasa a X-a

- implementarea unor programe personalizate folosind funcționalitățile clasei dict, pentru gestiunea stocurilor dintr-un magazin
- implementarea unor programe personalizate folosind funcționalitățile clasei dict, pentru traducerea automată a unor cuvinte în mai multe limbi
- 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

CS 6.5. Implementează aplicații în limbaj de programare care integrează subprograme recursive personalizate, pentru a modulariza și organiza eficient codul în cadrul soluțiilor informatice

- crearea unei aplicații pentru prelucrarea cifrelor unui număr care afișează mai multe opțiuni, iar pentru fiecare opțiune se apelează câte un subprogram recursiv (de exemplu, pentru afișarea cifrelor în ordine de la stânga la dreapta, de la dreapta la stânga, determinarea cifrei maxime, a numărului de cifre, a sumei cifrelor pare, verificarea existenței unei cifre pare, verificarea proprietății de platou format cu cifre egale), fiecare subprogram fiind elaborat de câte un membru al unei echipe de elevi
- crearea unei aplicații cu subprograme recursive pentru prelucrarea unei liste ordonate de numere (de exemplu, mediile unor elevi), prin care să se afișeze numerele atât în ordine crescătoare, cât și în ordine descrescătoare
- crearea unui subprogram recursiv pentru a verifica dacă un cuvânt este o anagramă a unui alt cuvânt, în contextul unei aplicații de jocuri enigmatice
- crearea unei aplicații cu subprograme recursive care să afișeze pe ecran diverse forme geometrice simple cu ajutorul spațiilor și al unor simboluri (de exemplu, un triunghi format din simboluri *), în contextul unui concurs de realizare a designului unei sigle

CONȚINUTURI ALE ÎNVĂȚĂRII

Clasa a X-a

Domenii de conținut	Conținuturi
1. Organizarea conceptuală a datelor	1.1. Modelul conceptual neliniar - mulțime - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.2. Modelul conceptual liniar – șir de caractere - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.3. Modelul conceptual asociativ – dicționar - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.4. Modelul conceptual mixt (de exemplu, liste de liste, liste de șiruri de caractere) - 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 - caracteristici ale metodei căutării binare și repere pentru aplicarea acesteia; - caracteristici ale metodei interclasării și repere pentru aplicarea acesteia. 2.2. Metoda Divide et impera - caracteristici ale metodei Divide et impera; - repere pentru aplicarea metodei. 2.3. Metoda Greedy - caracteristici ale metodei Greedy; - repere pentru aplicarea metodei.
3. Memorarea datelor și organizarea codului în limbaj de programare	3.1. Clasa set din Python – clasă predefinită pentru mulțimi - caracteristici; - operații și operatori specifici: reuniune, intersecție, diferență, egalitate, incluziune, apartenență; - metode uzuale pentru adăugare, ștergere, inițializare; - repere pentru identificarea și utilizarea altor metode adecvate pentru prelucrarea mulțimilor, în funcție de nevoile proprii. 3.2. Clasa str din Python – clasă predefinită pentru șiruri de caractere - caracteristici; - operații și operatori specifici: apartenență, concatenare, multiplicare, acces la un caracter sau subșir/secvență de caractere, comparare lexicografică; - 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; - repere pentru identificarea și utilizarea altor metode și funcții adecvate pentru prelucrarea șirurilor de caractere, în funcție de nevoile proprii. 3.3. Clasa dict din Python – clasă predefinită pentru dicționare - caracteristici; - operații și operatori specifici: acces la date; - repere pentru identificarea și utilizarea altor metode sau operatori adecvați pentru prelucrarea dicționarelor, în funcție de nevoile proprii. 3.4. Elemente de limbaj pentru operarea cu date organizate în modele mixte - caracteristici, acces la un element în Python. 3.5. Subprograme recursive - caracteristici, rol, mecanism de executare; - repere pentru scrierea subprogramelor recursive în Python.

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 măsoară? 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.

Un context de învățare poate viza simultan mai multe competențe, mai ales atunci când acestea sunt interconectate. De exemplu, pentru rezolvarea unei probleme cu ajutorul calculatorului se vizează, în cadrul aceleiași activități de învățare, atât competențe de nivel cognitiv scăzut (identificarea principalelor caracteristici ale modelelor conceptuale și operaționale implicate, explicarea principiilor care stau la baza acestora, în contextul particular al problemei, utilizarea lor în rezolvare) dar și competențe de nivel cognitiv ridicat (analizarea caracteristicilor modelelor conceptuale și operaționale specifice problemei, evaluarea corectitudinii și eficienței soluțiilor propuse și elaborarea algoritmului personalizat).

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, algoritimizarea, 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 metode/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, urmărirea 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 matematică-informatică:

1. *Strategii de rezolvare a problemelor. Metode de prelucrare a listelor ordonate*
2. *Organizarea conceptuală a datelor. Modelul conceptual neliniar - mulțime*
3. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa set din Python*
4. *Organizarea conceptuală a datelor. Modelul conceptual liniar – șir de caractere*
5. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa str din Python*
6. *Organizarea conceptuală a datelor. Modelul conceptual asociativ – dicționar*
7. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa dict din Python*
8. *Organizarea conceptuală a datelor. Modelul conceptual mixt*
9. *Memorarea datelor și organizarea codului în limbaj de programare. Subprograme recursive*
10. *Strategii de rezolvare a problemelor. Metoda Divide et impera*
11. *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 conceptului de mulțime reprezintă o punte naturală între gândirea matematică și cea informatică. Se recomandă utilizarea strategiei descoperirii dirijate, pornind de la situații cotidiene, de exemplu, „mulțimea elevilor din clasă”, „mulțimea culorilor preferate”, pentru a stimula înțelegerea noțiunii de element și apartenență. Se pot folosi diagrame Venn pentru reprezentarea relațiilor de reuniune, intersecție și diferență, susținând astfel dezvoltarea gândirii vizuale și logice. Se recomandă integrarea metodei comparației pentru a reliefa caracterul neliniar al acestei structuri față de listele ordonate. Profesorul concretizează teoria mulțimilor prin clasa predefinită set. Se recomandă demonstrația dirijată și învățarea activă: elevii experimentează singuri comportamentul metodelor (add, remove, union, intersection, difference). Se valorifică interdisciplinaritatea cu matematica, punând accent pe semnificația logică a operațiilor.

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.

Profesorul poate introduce modelul asociativ, reprezentat de dicționar, prin metoda modelării, explicând principiul perechilor cheie-valoare și apelând la exemple familiare (agenda de contacte, glosar de termeni). Elevii sunt ghidați prin conversație euristică și comparare între listă și dicționar, pentru a înțelege diferențele privind accesul la date. Se recomandă integrarea live coding-ului și a vizualizării în timp real a modificării structurilor, consolidând astfel legătura între teorie și practică.

Predarea modelelor mixte de organizare a datelor (liste de liste, liste de șiruri de caractere, liste de dicționare etc.) poate fi abordată gradual, pornind de la exemple simple și concrete (liste de liste, liste de dicționare), în care apare nevoia de ierarhizare și acces multiplu la date, către generalizări (dicționare de dicționare). 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.

Profesorul poate introduce conceptul de recursivitate folosind metoda analogiei („oglinda într-o oglindă”) și reprezentarea grafică a apelurilor recursive. Este utilă învățarea prin simulare, pentru a vizualiza mecanismul stivei de apeluri. Metacogniția are un rol major: elevii trebuie să descrie verbal condițiile de oprire și fluxul execuției.

Pentru metoda Divide et impera, profesorul poate să introducă ideea de descompunere a problemei prin problematizare și euristică. Se recomandă construirea treptată a conceptului prin exemple familiare (găsirea elementului maxim, căutarea unui element, sortarea unei liste). Diagramele arborescente și vizualizarea recursivității facilitează înțelegerea structurii logice a algoritmului. O strategie eficientă este metacogniția – elevii sunt invitați să explice verbal modul de împărțire și combinare a subproblemelor. La predare, se evidențiază aplicarea acestei strategii și pentru metoda căutării binare, cu eliminarea uneia dintre subproblemele generate, având în vedere că nu influențează rezultatul. Se recomandă abordarea unor rezolvări cu aplicarea algoritmilor de bază (de exemplu, sume, minime), probleme clasice (de exemplu, problema turnurilor din Hanoi, problema covorului lui Sierpinski) ca model, apoi realizarea unor probleme asemănătoare care presupun adaptări, reinterpretări, extinderi.

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.

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 limbi (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 Moisi”, 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
NIȚU Alina	profesor, gradul didactic I	Liceul Teoretic „Ovidius”, Constanța, județul Constanța

Nume și prenume	Grad didactic/Titlu științific	Instituție de apartenență, localitate, județ
NURLA Aidan	profesor, gradul didactic I	Colegiul Național Militar „Alexandru Ioan Cuza”, Constanța, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forțelor 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 Național Militar „Dimitrie Cantemir”, Breaza, județul Prahova
TEGLAȘ Maria	profesor, gradul didactic II	Colegiul Național 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

Anexa nr. 3

Programa școlară
pentru disciplina

Informatică

Clasa a XI-a

Curriculum de specialitate (CS)

Filiera teoretică, profilul real, specializarea matematică-informatică

Învățământ liceal

- 2026 –

Anexa nr. 3**Programa școlară pentru disciplina Informatică, clasa a XI-a, curriculum de specialitate (CS)**

- *Filiera teoretică, profilul real, specializarea matematică-informatică*

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 matematică-informatică, alocarea orară este:

- clasa a IX-a – 2 ore/săptămână, dintre care o oră pentru studiu teoretic și o oră pentru activități practice;
- clasa a X-a – 2 ore/săptămână, dintre care o oră pentru studiu teoretic și o oră pentru activități practice;
- clasa a XI-a – 4 ore/săptămână, dintre care două ore pentru studiu teoretic și două ore pentru activități practice;
- clasa a XII-a – 3 ore/săptămână, dintre care o oră pentru studiu teoretic și două ore pentru activități practice.

Activitățile practice sunt desfășurate obligatoriu în laboratorul de informatică.

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** (CDEOS), 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 atestatorilor de certificare a competențelor profesionale sau pentru satisfacerea unor nevoi și interese pentru specializare la nivel superior, inclusiv pentru admiterea la studii universitare.

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.

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 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 asigură o curbă de învățare blândă și 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 XI-a
CS 1.1. Identifică principalele caracteristici ale organizării datelor în cadrul unor modele conceptuale complexe – rețea, ierarhice, obiectuale, pentru a structura și accesa date în vederea prelucrării acestora
- definirea conceptului de graf, cu exemplificarea reprezentării rutelor directe bidirecționale dintre anumite localități aflate pe harta rutieră a unei țări, pentru un graf neorientat, respectiv cu exemplificarea reprezentării străzilor cu sens unic sau cu circulație pe ambele sensuri, aflate pe harta rutieră a unei localități, pentru un graf orientat - recunoașterea caracteristicilor grafurilor ponderate (orientate sau neorientate), prin completarea unei fișe comparative, pentru exemplificarea reprezentării unei hărți rutiere a unui oraș care are și străzi unidirecționale, respectiv a unei regiuni cu șosele bidirecționale, în care costurile arcelor/muchiiilor indică distanța sau timpul de deplasare între intersecțiile/localitățile corespunzătoare - identificarea unor noduri într-un arbore cu rădăcină (nod descendent direct, nod descendent, nod ascendent direct, nod ascendent, nod frate) pentru reprezentarea relațiilor ierarhice într-o organizație - identificarea metodelor de reprezentare a arborilor, cu referințe ascendente sau referințe descendente, pentru exemplificarea reprezentării relațiilor de rudenie dintre strămoși și descendenți sub forma unui arbore genealogic limitat la linia paternă - identificarea caracteristicilor unei clase, prin exemplul stabilirii membrilor clasei „Rețea”, care încapsulează datele tehnice și metodele de întreținere a unei rețele - identificarea caracteristicilor moștenirii, punând în evidență „specializarea” unei clase, de exemplu pentru clasele „Autoturism” și „Camion” care sunt derivate din clasa „Vehicul”
CS 1.2. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea grafurilor, arborilor
- precizarea etapelor algoritmului de parcurgere în lățime (BFS), în contextul explorării unei rețele de comunicații, unde algoritmul simulează verificarea tuturor legăturilor de pe același nivel - precizarea etapelor algoritmului de parcurgere în adâncime (DFS), în contextul analizării accesibilității nodurilor unei rețele de comunicații, pe care algoritmul le explorează în profunzime - definirea scopului algoritmului lui Prim, indicând tipul de graf pentru care se aplică, în contextul utilizării acestuia pentru analizarea asfaltării unor străzi dintr-un oraș, astfel încât, cu un cost minim de asfaltare, să existe posibilitatea deplasării pe asfalt între oricare două intersecții, eventual trecând pe asfalt și prin alte intersecții - enumerarea modalităților de parcurgere a arborilor binari: preordine și postordine, în contextul unei structuri ierarhice, în care fiecare angajat are cel mult doi subalterni, unde parcurgerea în preordine reflectă transmiterea mesajelor de la cel care conduce spre subordonați, iar postordine – raportarea rezultatelor de jos în sus - precizarea etapelor operațiilor specifice arborelui binar de căutare, în contextul inserării și căutării informațiilor unor dispozitive dintr-o rețea de comunicații, identificate prin coduri, în cadrul unui catalog
CS 1.3. Identifică specificul, caracteristicile și etapele unor strategii de tip Backtracking pentru rezolvarea problemelor
- precizarea caracteristicilor metodei de programare Backtracking, în contextul modelării procesului decizional pentru construirea unui robot, în limita unui buget dat, unde se explorează toate opțiunile posibile de alegere a pieselor necesare, cu testarea condițiilor de continuare și eliminarea ramurilor invalide, revenindu-se la starea anterioară în momentul în care pentru starea curentă au fost explorate toate variantele - identificarea caracteristicilor metodei Backtracking, în contextul generării unor liste de obiective turistice care să fie vizitate succesiv într-un traseu turistic, respectând anumite restricții (durată, conexiuni rutiere etc.) - identificarea principiului de generare sistematică a secvențelor de valori în cadrul metodei Backtracking, în contextul testării succesive a tuturor variantelor de amplasare a unor camere web de observare pe unii dintre stâlpii unei străzi pentru acoperirea completă a acesteia
CS 1.4. Identifică principalele elemente ale limbajului de programare utilizate pentru memorarea și prelucrarea datelor organizate în modele complexe - rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice
- identificarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat sau orientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare - identificarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu liste de adiacență, respectiv a unui graf orientat, cu liste de succesori, în contextul parcurgerii nodurilor acestuia

Clasa a XI-a
- identificarea instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore binar, astfel încât să permită accesul la fiii oricărui nod
CS 1.5. Identifică elementele de sintaxă și componentele din definiția claselor, pentru a le utiliza în implementarea algoritmilor în limbaj de programare cu programare orientată pe obiecte
- identificarea convențiilor de numire a membrilor unei clase (public, privat) și sintaxa prin care sunt stabilite acestea, pentru a recunoaște controlul vizibilității, prin exemplul definirii unei clase „Rețea” unde atributele tehnice sunt private, iar metodele de întreținere sunt publice, pentru a reflecta controlul accesului la date sensibile - identificarea unor elemente de sintaxă pentru definirea unei clase, de exemplu, pentru clasa „Vehicul” cu atribute precum tip, viteză, stoc combustibil și metode pentru actualizarea vitezei sau realimentare - identificarea sintaxei de definire a constructorului unei clase, de exemplu, pentru inițializarea datelor unui utilizator pe o platformă de cursuri online, precum și a drepturilor pe care le are pe aceasta

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 XI-a
CS 2.1. Explică principiile de organizare a datelor în cadrul unor modele conceptuale complexe - rețea, ierarhice, obiectuale, pentru a le gestiona eficient
- explicarea particularităților metodelor de reprezentare a grafurilor (matrice de adiacență, liste de adiacență), în contextul reprezentării conexiunilor dintre localitățile identificate pe o hartă rutieră, respectiv a unei rețele logistice de alimentare cu apă a unei localități care conectează depozitele de apă cu dispozitivele de pompare a apei în rețea - explicarea rolului elementelor componente ale unei diagrame prin care se reprezintă un graf orientat, respectiv un graf neorientat prin exemplul relațiilor dintre membrii unui grup de persoane: în cazul în care doar unii sunt populari, cunoscuți și de alți membri, chiar dacă ei nu îi cunosc pe aceștia (graf orientat) respectiv în cazul unor relații de prietenie reciprocă (graf neorientat) - explicarea relațiilor de descendență și ascendență într-un arbore cu rădăcină, prin exemplificarea cu arbori genealogici, cu limitare la linia paternă, în care ordinea descendenților nu este evidențiată, unde părinții și bunicii reprezintă relațiile de ascendență, iar copiii și nepoții reprezintă relațiile de descendență - explicarea relațiilor ierarhice într-un arbore cu rădăcină, pe baza exemplului unei organizații în care fiecare angajat, cu excepția celor cu funcții de execuție, are colegi care îi sunt direct subordonați - explicarea noțiunii de cheie și a condiției de ordine într-un arbore binar de căutare, prin exemplul utilizării unui criteriu de identificare (precum un cod unic) pentru indexarea și căutarea informațiilor într-un sistem de comunicații, unde fiecare cheie/cod are o poziție determinată în funcție de valoarea sa - explicarea încapsulării, prin exemplul stabilirii membrilor unei clase, „Rețea”, unde se dorește ca datele tehnice să nu fie accesibile direct pentru utilizator, pentru a reflecta controlul asupra datelor sensibile - explicarea moștenirii, punând în evidență „specializarea” unei clase, de exemplu pentru clasele „Autoturism” și „Camion” care sunt derivate din clasa „Vehicul”
CS 2.2. Explică etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea grafurilor, arborilor
- explicarea etapelor de funcționare ale algoritmului de parcurgere în lățime (BFS) a grafurilor, prin exemplul identificării rutei optime de comunicare între punctele de distribuție a unui produs (depozite) și punctele de desfacere (magazine), analizând ordinea vizitării nodurilor (depozite și magazine) și a legăturilor dintre acestea - explicarea etapelor de funcționare ale algoritmului lui Dijkstra de determinare a unui drum de cost minim într-un graf, prin exemplul identificării rutei optime de transport între două puncte de distribuție a unui produs (depozite), având în vedere alegerea nodurilor (depozitelor) și actualizarea adecvată a distanțelor rutiere corespunzătoare legăturilor dintre acestea - explicarea etapelor algoritmului lui Prim, prin exemplul construirii unei rețele de comunicații cu un cost total minim, integrând, după caz, pas cu pas, conexiunile cu costul cel mai mic - explicarea metodelor de parcurgere a arborilor binari, prin exemplul ilustrării modului în care datele despre organizarea unei excursii sunt procesate în ordine diferită, în funcție de scop: planificarea etapelor principale și a detaliilor (preordine) sau evaluarea și centralizarea costurilor finale (postordine)
CS 2.3. Explică etapele unor strategii de tip Backtracking pentru rezolvarea problemelor
- explicarea modului de generare a tuturor soluțiilor utilizând metoda Backtracking, în contextul procesului de planificare a rutelor unei companii de transport aerian, unde fiecare destinație reprezintă o decizie dependentă de destinația anterioară - explicarea rolului validării pe parcursul determinării soluțiilor cu metoda Backtracking, în contextul verificării condițiilor (acoperire, vizibilitate, siguranță) de amplasare a unor senzori într-o rețea de comunicații, în toate modurile posibile

Clasa a XI-a	
-	<i>explicarea procesului de revenire la starea anterioară pentru generarea unei soluții cu metoda Backtracking, prin exemplul epuizării tuturor variantelor de alegere a unui dispozitiv adecvat pentru supravegherea unei anumite zone și necesitatea de a le reconfigura pe cele deja plasate</i>
CS 2.4. Explică rolul principalelor elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele complexe - rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice	
-	<i>explicarea rolului instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor dintr-un grup de persoane, în care sunt evidențiate relațiile de prietenie</i>
-	<i>explicarea modului în care pot fi utilizate instrumentele clasei list pentru reprezentarea în memorie a unui graf orientat, cu liste de predecesori, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de transmitere a unor informații, cu precizarea sursei</i>
-	<i>explicarea rolului instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore binar, astfel încât să permită accesul la fiii oricărui nod, în contextul unui arbore corespunzător unui chestionar, în care fiecare întrebare reprezintă un nod și este urmată, eventual, de o altă întrebare, în funcție de răspunsul dat (corect, incorect)</i>
CS 2.5. Explică rolul restricțiilor de acces și al componentelor unei clase, pentru a le utiliza în proiectarea claselor și implementarea algoritmilor cu programare orientată pe obiecte	
-	<i>explicarea modului în care constructorul inițializează starea unui obiect, pentru a crea corect instanțele, în contextul unui obiect „Concurs” (locație, disciplină, echipă) la crearea acestuia</i>
-	<i>explicarea modului de utilizare a membrilor clasei pentru a înțelege manipularea obiectelor într-un program, prin exemplul utilizării metodelor din clasa „Rețea” pentru a gestiona dispozitivele din rețea</i>
-	<i>explicarea principiului de moștenire a unei clase pentru reutilizarea și extinderea codului, prin exemplul clasei „Angajat”, extinsă prin clase derivate precum „Inginer” și „Profesor”, care moștenesc membrii comuni, dar adaugă membri specifici</i>
-	<i>explicarea relației de moștenire între clase pentru a interpreta reutilizarea și extinderea codului prin exemplul unei clase pentru autovehicule care transportă persoane și clasele derivate pentru autoturisme, respectiv pentru autocare</i>

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

Clasa a XI-a	
CS 3.1. Utilizează modul de organizare și prelucrare a datelor în cadrul unor modele conceptuale complexe - rețea, ierarhice, obiectuale, pentru a rezolva probleme de gestionare a datelor	
-	<i>utilizarea matricei de adiacență pentru a reprezenta un graf neorientat dat prin diagramă, în contextul modelării unei rețele de calculatoare conectate într-un laborator de informatică, evidențiind prezența valorilor nule pe diagonala principală, simetria față de aceasta</i>
-	<i>utilizarea listelor de succesori pentru a reprezenta un graf orientat dat prin matrice de adiacență, în contextul modelării fluxului de transmitere a datelor într-o rețea de comunicații</i>
-	<i>utilizarea matricei de adiacență pentru a reprezenta un graf orientat dat prin diagramă în contextul modelării căilor de deplasare a autovehiculelor pe străzile unui oraș, având în vedere că acestea pot fi cu sens unic și că unesc intersecții, evidențiind prezența valorilor nule pe diagonala principală, asimetria față de aceasta</i>
-	<i>aplicarea metodei de reprezentare cu referințe descendente pentru un arbore dat prin diagramă, în contextul construirii unui arbore cu rădăcină pentru a reprezenta o structură organizatorică simplificată, cu niveluri corespunzătoare pentru diferite funcții</i>
-	<i>aplicarea metodei de reprezentare prin diagramă pentru un arbore dat prin referințe ascendente, în contextul determinării numărului de niveluri ale unui arbore cu rădăcină ce reprezintă o structură organizatorică simplificată, cu niveluri corespunzătoare pentru diferite funcții</i>
-	<i>aplicarea metodei de reprezentare cu referințe descendente (fii) pentru un arbore binar dat prin diagramă, în contextul determinării frunzelor acestuia</i>
-	<i>aplicarea încapsulării, prin exemplul stabilirii membrilor unei clase, „Rețea”, unde datele tehnice și metodele de întreținere caracterizează o rețea de calculatoare</i>
-	<i>aplicarea moștenirii, punând în evidență „specializarea” unei clase, de exemplu pentru clasele „Autoturism” și „Camion” care sunt derivate din clasa „Vehicul”</i>
CS 3.2. Aplică algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea grafurilor, arborilor	
-	<i>aplicarea algoritmilor BFS și DFS, pentru parcurgerea grafurilor neorientate, pe baza unei reprezentări prin diagramă a unui exemplu de graf neorientat, în care se marchează nodurile vizitate și se evidențiază particularitățile celor doi algoritmi</i>
-	<i>aplicarea algoritmului lui Dijkstra pentru a determina distanța minimă între două vârfuri, într-un graf orientat dat, prin exemplul identificării traseului optim între două orașe ținând cont de costul deplasării</i>

Clasa a XI-a	
- aplicarea algoritmului lui Prim pentru a determina arborele de acoperire de cost minim într-un graf neorientat ponderat dat, prin exemplul proiectării unei rețele de transport care conectează toate sediile unei companii, cu un cost total minim de transport - aplicarea metodelor de parcurgere (inordine, preordine, postordine) a nodurilor unui arbore binar, dat printr-o diagramă, evidențiind particularitățile fiecăreia și poziția, în șirul obținut, a rădăcinii arborelui/subarborelui pentru fiecare caz - aplicarea algoritmului de căutare a unei valori într-un arbore de căutare, prin exemplul căutării unui cuvânt într-un grup de cuvinte organizate sub forma unui arbore binar de căutare, evidențiindu-se particularitățile acestuia și numărul de comparații necesare	
CS 3.3. Aplică strategii de tip Backtracking pentru rezolvarea problemelor	
- aplicarea etapelor metodei Backtracking pentru generarea tuturor permutărilor unei mulțimi de patru cifre date, într-un context cotidian, prin exemplul generării tuturor codurilor posibile de câte 4 cifre distincte, pentru deblocarea unui dispozitiv, știind că utilizatorul își amintește că ultimele două cifre sunt pare - aplicarea etapelor metodei Backtracking pentru generarea tuturor combinațiilor unui set de obiecte într-un context cotidian, prin exemplul determinării tuturor posibilităților de a forma grupe de câte un număr fixat de echipamente necesare pentru un concurs de robotică - aplicarea etapelor metodei Backtracking pentru generarea sistematică a unor secvențe de valori, determinând eficient ultimele două soluții generate, prin exemplul planificării diferitelor scenarii de deplasare a unor drone care să includă toate punctele de interes	
CS 3.4. Utilizează principalele elemente ale limbajului de programare pentru prelucrarea datelor organizate în modele complexe - rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice	
- utilizarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de pe o hartă, în care sunt evidențiate relațiile de vecinătate dintre țări - utilizarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf orientat, cu liste de succesori, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de transmitere a unor informații, cu precizarea destinației - utilizarea instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la tatăl oricărui nod, în contextul unui arbore asociat unui sistem de fișiere în care rădăcina reprezintă directorul principal, iar fiecare folder (sau fișier) are un singur director părinte	
CS 3.5. Utilizează elementele de sintaxă și componentele din definiția claselor, în implementarea algoritmilor în limbaj de programare cu programare orientată pe obiecte	
- utilizarea elementelor de sintaxă ale limbajului de programare pentru definirea unei clase cu atribute și metode deja stabilite, pentru gestionarea unei situații reale, prin exemplul proiectării unei clase „ActivitateScolara” care gestionează datele despre activitățile dintr-un proiect școlar, de exemplu, codul activității, denumirea, obiectivul principal, numărul participanților etc. - utilizarea elementelor de sintaxă pentru crearea unei clase cu un constructor și metode pentru gestionarea unei situații reale, prin exemplul implementării unei clase „Autoturism” care include un constructor pentru inițializare (marcă, model, an de fabricație) și metode pentru îmbunătățirea performanțelor (de exemplu, tuning pentru motor), modificare a culorii - utilizarea elementelor de sintaxă pentru crearea unei clase care utilizează date ale unor clase definite anterior, de exemplu, clasa „Rezolvare” are drept membri obiecte din clasele „Elev”, respectiv „Problema”, asigurând corelarea dintre problemele propuse și elevii care le-au rezolvat - utilizarea elementelor de sintaxă pentru a defini clase derivate prin exemplul obținerii, din clasa de bază „JocVideo”, a claselor derivate „JocEducational” și „JocStrategie”, în scopul reutilizării componentelor comune și al extinderii acestora cu atribute și metode specifice fiecărui tip de joc	

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 XI-a	
CS 4.1. Analizează caracteristicile, modul de prelucrare, avantajele și dezavantajele utilizării unor modele conceptuale complexe - rețea, ierarhice, obiectuale, pentru a alege structura adecvată în rezolvarea unor probleme concrete	
- analizarea caracteristicilor unui graf neorientat, conex, în contextul examinării traseului mașinii unei echipe care întreține indicatoarele rutiere din toate intersecțiile unei localități, pentru a verifica dacă mașina le poate accesa pe toate - examinarea gradelor nodurilor unui graf orientat asociat unei rețele de străzi dintr-un oraș pentru a identifica intersecțiile conectate la un număr mare de străzi, în vederea fluidizării traficului din localitate - analizarea reprezentării prin diagramă a unui graf orientat asociat unei rețele de comunicații, în contextul stabilirii direcțiilor de transmitere a semnalelor - analizarea definițiilor echivalente ale unui arbore, în contextul rezolvării unei probleme de natură informatică, în care se verifică organizarea datelor sub formă de arbore, pentru a o alege pe cea mai ușor de implementat	

Clasa a XI-a	
- <i>analizarea relațiilor ierarhice în arborii cu rădăcină (înălțime, noduri descendente, noduri ascendente), în contextul comparării mai multor arbori asociați sistemului de foldere și subfoldere din calculator</i> - <i>analizarea diferențelor dintre metodele de reprezentare a unui arbore binar, având în vedere relațiile părinte-copil și proprietățile arborelui</i> - <i>analizarea unei structuri pentru a determina dacă are caracteristici specifice arborelui binar de căutare asociat organizării datelor elevilor unei clase, pentru a asigura accesul rapid și ordonat la informații, în funcție de numerele matricole ale elevilor</i> - <i>analizarea membrilor unei clase pentru a verifica dacă aceasta conține toate datele și metodele necesare pentru a răspunde unei funcționalități, prin exemplul clasei „Rețea”, care cuprinde datele tehnice și metodele de întreținere a unei rețele de calculatoare</i> - <i>analizarea oportunității de moștenire, punând în evidență „specializarea” unei clase, de exemplu pentru clasele „Autoturism” și „Camion” care sunt derivate din clasa „Vehicul”</i>	
CS 4.2. Analizează comportamentul, aplicabilitatea, avantajele și dezavantajele utilizării unor algoritmi specializați pe clase de probleme pentru prelucrarea grafurilor, arborilor	
- <i>analizarea metodelor BFS și DFS de parcurgere a grafurilor în vederea utilizării lor pentru verificarea proprietății de conexitate a unui graf neorientat, prin exemplul identificării componentelor izolate dintr-o rețea de comunicații bidirecționale, pentru a determina întreruperi în lanțul de transmitere a datelor</i> - <i>analizarea metodelor BFS versus DFS de parcurgere a unui graf neorientat/orientat în vederea adaptării lor pentru determinarea lanțului/drumului cu lungime minimă între două noduri ale grafului, respectiv pentru determinarea unui drum/lanț între două noduri ale grafului</i> - <i>analizarea rezultatelor parcurgerii în lățime a nodurilor unui graf neorientat, cu interpretarea ordinii nodurilor afișate, prin exemplul comparării ordinii de procesare a informațiilor într-un sistem de comunicații</i> - <i>analizarea complexității algoritmului Dijkstra, prin exemplul evaluării performanței sistemului de planificare a rutelor unui traseu turistic tematic, în care se cunosc distanțele dintre obiective, în funcție de numărul de puncte de destinație și de conexiuni disponibile</i> - <i>analizarea rezultatelor algoritmului lui Prim, în contextul determinării acoperirii de cost minim a unui graf ponderat, prin exemplul construirii unui model grafic care indică legăturile logistice de bază necesare pentru conectarea tuturor sediilor unei firme de transport</i> - <i>compararea metodelor de parcurgere a arborilor binari pentru a determina situațiile optime de utilizare a acestora prin exemplul alegerii metodei de parcurgere adecvate într-un sistem de planificare a activităților</i> - <i>analizarea modului în care inserarea unui nod afectează structura generală a unui arbore binar de căutare prin exemplul inserării unor elevi care s-au alăturat colectivului clasei</i> - <i>analizarea efectului schimbării ordinii de inserare a cheilor într-un arbore binar de căutare, prin exemplul examinării modului în care această ordine afectează structura arborelui</i>	
CS 4.3. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de tip Backtracking pentru rezolvarea problemelor	
- <i>analizarea mecanismelor de avans și de revenire în arborele soluțiilor la aplicarea metodei Backtracking, prin exemplul generării tuturor anagramelor cu anumite proprietăți ale unui cuvânt format cu litere distincte (de exemplu, astfel încât literele să nu fie alăturate și în cuvântul dat), pentru evidențierea modului în care algoritmul explorează adecvat posibilitățile și revine pentru a face alte alegeri de litere, în vederea obținerii de noi soluții</i> - <i>analizarea avantajelor și dezavantajelor aplicării metodei Backtracking comparativ cu alte metode de generare pentru a evalua eficiența acestora în obținerea tuturor numerelor naturale cu anumite proprietăți de aranjare a cifrelor, în funcție de numărul acestora</i> - <i>analizarea aplicabilității metodei Backtracking, în funcție de numărul de soluții posibile, prin exemplul evaluării timpului de executare pentru planificarea tuturor rutelor aeriene pentru mai multe avioane care zboară simultan</i>	
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 complexe - rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice	
- <i>analizarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență sau cu liste de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de identificare a calculatoarelor dintr-o rețea, în care sunt evidențiate relațiile de conexiune directă dintre acestea</i> - <i>analizarea instrumentelor clasei list pentru reprezentarea în memorie, cu liste de predecesori, a unui graf orientat corespunzător unui sistem de transmitere a unor informații, în care vârfurile reprezintă stații de emisie-recepție și pentru fiecare se precizează sursele de la care primește direct informații, în contextul determinării, pentru o stație dată, a tuturor surselor directe sau indirecte de informații</i> - <i>analizarea instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la tatăl oricărui nod, în contextul unui arbore asociat unui sistem ierarhic din cadrul unei organizații în</i>	

Clasa a XI-a
<i>care rădăcina corespunde directorului general, fiecare alt angajat are un singur manager direct, iar în organizație nu există cicluri de subordonare</i>
CS 4.5. Analizează aplicabilitatea, avantajele și dezavantajele utilizării programării orientate pe obiecte, în implementarea algoritmilor în limbaj de programare
- <i>analizarea avantajelor și dezavantajelor utilizării convențiilor de numire a membrilor unei clase pentru a interpreta impactul asupra securității datelor, prin exemplul protejării datelor sensibile într-o aplicație de evidență a angajaților dintr-o instituție (date de identificare, adresă de domiciliu)</i> - <i>analizarea unei aplicații cu mai multe clase pentru a identifica dependențele și interacțiunile dintre obiecte, prin exemplul analizării relațiilor dintre clasele „Carte”, „Cititor”, „Biblioteca”, pentru o bibliotecă publică</i>

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

Clasa a XI-a
CS 5.1. Argumentează alegerea unor modele conceptuale complexe - rețea, ierarhice, obiectuale și a modurilor de prelucrare a acestora pentru rezolvarea unor probleme informatice concrete
- <i>argumentarea alegerii metodelor de reprezentare a grafurilor neorientate prin matrice de adiacență sau liste de adiacență, în contextul alegerii modului optim de reprezentare a unei rețele de transport aerian, având în vedere numărul destinațiilor și densitatea conexiunilor</i> - <i>evaluarea alegerii modelării printr-un graf orientat a unei situații reale, prin exemplul validării unei diagrame de proces care descrie legăturile condiționale (dependențe) dintre etapele unei activități</i> - <i>argumentarea alegerii unei metode de reprezentare a arborilor cu rădăcină (referințe descendente, referințe ascendente), în funcție de scopul utilizării, prin exemplul alegerii metodei optime de reprezentare a structurii unei echipe care va participa la un concurs de robotică, în funcție de necesitatea de prezentare ierarhică a componenței echipei sau de detaliere a rolului fiecărui membru</i> - <i>argumentarea utilizării unui arbore binar de căutare în contextul căutării unor informații într-o mulțime dată, în mod repetat</i> - <i>argumentarea alegerii membrilor unei clase, astfel încât aceasta să conțină toate datele și metodele necesare pentru a răspunde unei funcționalități, prin exemplul clasei „Rețea”, care cuprinde datele tehnice și metodele de întreținere a unei rețele de calculatoare</i> - <i>argumentarea alegerii moștenirii, punând în evidență „specializarea” unei clase, de exemplu pentru clasele „Autoturism” și „Camion” care sunt derivate din clasa „Vehicul”</i>
CS 5.2. Evaluează corectitudinea și eficiența soluțiilor cu algoritmi specializați pe clase de probleme pentru prelucrarea grafurilor, arborilor
- <i>evaluarea menținerii proprietății de conexitate utilizând metodele de parcurgere a unui graf neorientat în contextul unei rețele de comunicații având în vedere posibilitatea menținerii legăturii între dispozitive în caz de pierdere a unui nod (nod critic), prin înlăturarea unui dispozitiv</i> - <i>evaluarea eficienței algoritmului lui Dijkstra pentru determinarea drumurilor de cost minim într-un graf ponderat, prin exemplul determinării rutelor optime de aprovizionare între un depozit și punctele de vânzare</i> - <i>evaluarea eficienței algoritmului lui Prim pentru determinarea acoperirii de cost minim într-un graf ponderat, prin exemplul determinării rețelei logistice de transport cu cost minim</i> - <i>evaluarea corectitudinii alegerii uneia dintre metodele de parcurgere ale arborilor binari, în contextul specificului unui sistem ierarhic: preordine pentru diseminarea rapidă a sarcinilor, postordine pentru evaluarea completă, inordine pentru generarea rapoartelor cu elemente sortate</i> - <i>evaluarea complexității algoritmului de căutare într-un arbore binar de căutare comparativ cu căutarea liniară, prin exemplul comparării timpului mediu necesar pentru regăsirea unui cod într-o structură ordonată după chei versus o listă nesortată</i>
CS 5.3. Evaluează corectitudinea și eficiența algoritmilor care utilizează strategii de tip Backtracking pentru rezolvarea problemelor
- <i>evaluarea corectitudinii aplicării metodei Backtracking pentru probleme de generare, prin exemplul determinării modului de parcurgere, într-o croazieră, a unor insule, astfel încât fiecare dintre acestea să fie vizitată o singură dată (cu excepția celei de plecare, la care se și încheie excursia), iar vasul să urmeze doar trasee acceptate între două insule vizitate consecutiv și să aducă turiștii, la final, în punctul de plecare</i> - <i>evaluarea corectitudinii aplicării metodei Backtracking pentru generarea tuturor rutelor de livrare, cu anumite restricții privind ordinea punctelor parcurse pe traseu, prin exemplul comparării eficienței explorării exhaustive a variantelor de traseu și verificarea finală a restricțiilor impuse, comparativ cu eliminarea pe parcurs a unor trasee necorespunzătoare</i> - <i>evaluarea complexității unui algoritm care folosește metoda Backtracking, prin exemplul estimării timpului necesar pentru explorarea tuturor variantelor de poziționare a unor echipamente video într-o rețea de comunicații</i>

Clasa a XI-a	
CS 5.4. Evaluează corectitudinea și eficiența aplicațiilor care utilizează elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele complexe - rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice	
- evaluarea aplicabilității instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de identificare a unor persoane dintr-un grup, evidențiind relațiile de prietenie dintre acestea - evaluarea instrumentelor clasei list pentru a le alege pe cele adecvate pentru reprezentarea în memorie a unui graf orientat, cu liste de succesori, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de transmitere a unor informații - argumentarea utilizării instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la descendenții direcți ai oricărui nod, în contextul unui arbore asociat unui sistem ierarhic din cadrul unei organizații în care rădăcina corespunde directorului general, unui angajați au mai mulți subordonați direcți, iar în organizație nu există cicluri de subordonare	
CS 5.5. Evaluează corectitudinea și eficiența programelor care utilizează programare orientată pe obiecte, pentru a rezolva probleme informatice	
- evaluarea corectitudinii definirii și utilizării claselor pentru organizarea datelor în contextul implementării unei aplicații de evidență a echipamentelor IT dintr-o școală, comparând varianta în care au fost definite clase față de organizarea cu funcții și variabile globale, din perspectiva clarității și modularității codului, cu avantaje pentru întreținerea acestuia și a prevenirii erorilor la actualizarea datelor - evaluarea corectitudinii aplicării convențiilor de numire pentru protecția informațiilor sensibile într-o aplicație reală, prin exemplul unei clase pentru gestionarea datelor personale ale elevilor, comparând accesul liber la date cu utilizarea unor date interne (protejate prin convenții) și a unor metode publice de acces, pentru validarea modificărilor și menținerea consistenței datelor - evaluarea corectitudinii definirii unei clase cu un constructor și metode pentru gestionarea unei situații reale, prin exemplul implementării unei clase „CursOnline” care include un constructor pentru inițializare (denumire, perioadă, număr de ore petrecute online, cost) și metode pentru actualizarea numărului de cursanți sau pentru modificarea perioadei - evaluarea corectitudinii definirii unei clase care are drept membri obiecte ale unor clase definite anterior, de exemplu, clasa „Optiune” are drept membri obiecte ale claselor „Client”, respectiv „Excursie”, asigurând corelarea dintre excursiile propuse și clienții care s-au înscris la acestea	

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

Clasa a XI-a	
CS 6.1. Proiectează algoritmi personalizați care utilizează modele conceptuale complexe - rețea, ierarhice, obiectuale, pentru organizarea și prelucrarea eficientă a datelor, utilizând algoritmi de bază, în vederea rezolvării unor probleme	
- proiectarea unui algoritm care utilizează grafuri neorientate pentru a gestiona datele referitoare la țările dintr-o regiune, determinând vecinii fiecăreia, țările cu cei mai mulți vecini sau țările izolate, înconjurate de ape internaționale în afara apelor teritoriale proprii - proiectarea unui algoritm care utilizează grafuri orientate în contextul modelării căilor de deplasare a autovehiculelor pe străzile unui oraș, având în vedere că acestea pot fi cu sens unic și că unesc intersecții, determinând străzile cu sens unic și punctele reprezentând extremități ale unor străzi fără ieșire/fundături - proiectarea unui algoritm pentru modelarea unui meniu arborecent, utilizând un arbore cu rădăcină în care fiecare nod corespunde opțiunii pentru o anumită acțiune, cu posibilitatea de a opta și pentru acțiuni subsecvente acesteia - proiectarea unui algoritm care utilizează arbori ordonați, determinând înălțimea arborelui, nodurile frunză, în contextul modelării unui arbore genealogic al unei familii limitat la linia paternă, cu descendenții direcți ordonați după vârstă - proiectarea unui algoritm care utilizează arborele binar de căutare pentru a identifica rapid datele dintr-un depozit de produse de papetărie pe baza codului unic de inventar - proiectarea unui algoritm pentru un model obiectual care încapsulează date și metode, prin exemplul determinării numărului total de exemplare ale publicațiilor dintr-o bibliotecă, în care o publicație este definită ca un model obiectual (titlu, autor, număr de exemplare și metode pentru modificarea numărului de exemplare), iar o bibliotecă este definită ca un model obiectual (listă de publicații, metode pentru adăugarea unei noi publicații) - proiectarea unui algoritm pentru un model obiectual cu clase derivate, punând în evidență utilizarea unor date și metode moștenite, precum și a unor date și metode specifice claselor derivate, de exemplu pentru clasele „Autoturism” și „Camion” derivate din clasa „Vehicul”	

Clasa a XI-a	
CS 6.2. Proiectează soluții cu aplicarea personalizată a algoritmilor specializați pe clase de probleme pentru prelucrarea grafurilor, arborilor	
- proiectarea unui algoritm care utilizează metoda de parcurgere în lățime a nodurilor unui graf prin exemplul unui centru de coordonare pentru situații de urgență care trebuie să trimită rapid echipe medicale către toate localitățile, pornind din orașul în care se află centrul, în situația unor intervenții - proiectarea unui algoritm care utilizează metoda de parcurgere în adâncime a nodurilor unui graf în contextul determinării unui traseu de la un punct de acces către un punct dat, printr-o rețea de tuneluri sau galerii miniere ale unei mine vechi - proiectarea unui algoritm pentru calcularea costurilor minime de livrare pentru o rețea de depozite, unități de desfacere și rute modelate cu un graf ponderat, utilizând algoritmul lui Dijkstra, prin exemplul optimizării transportului de la un depozit regional și unitățile de desfacere, pe baza distanțelor și costurilor de deplasare - proiectarea unui algoritm care utilizează algoritmul lui Prim prin exemplul proiectării unei rețele de iluminare stradală la care să fie conectați, prin cabluri, toți stâlpii de iluminat din oraș, iar lungimea totală a cablurilor utilizate să fie minimă - proiectarea unui algoritm care, pornind de la două șiruri de valori, reprezentând nodurile aceluiași arbore binar, parcurse în preordine, respectiv inordine, determină nodurile în ordinea parcurgerii lor în postordine - crearea unei aplicații pentru simularea căutării într-un arbore binar de căutare pentru identificarea unui elev într-o bază de date școlară, după criteriul mediei, iar la medii egale, alfabetic, după nume	
CS 6.3. Proiectează algoritmi cu aplicarea personalizată a strategiilor de tip Backtracking pentru rezolvarea problemelor	
- proiectarea unui algoritm care rezolvă o problemă folosind metoda Backtracking, pentru planificarea poziționării camerelor de supraveghere într-o clădire, având în vedere anumite restricții - proiectarea unui algoritm folosind metoda Backtracking pentru a determina toate modalitățile de a obține buchete de flori dintr-o mulțime dată, astfel încât acestea să aibă culori diferite	
CS 6.4. Implementează aplicații care integrează în mod personalizat elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele complexe - rețea, ierarhice, în vederea realizării unor soluții funcționale și performante	
- implementarea unei aplicații care utilizează clasa list pentru reprezentarea în memorie a unui graf orientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de identificare a unor persoane dintr-un grup, evidențiind modul în care o persoană cunoaște sau nu o altă persoană - implementarea unei aplicații care utilizează clasa list pentru reprezentarea în memorie a unui graf neorientat, cu liste de muchii, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de comunicare între prieteni - implementarea unei aplicații care utilizează instrumentele clasei list pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la ascendentul direct al oricărui nod, în contextul unui arbore asociat unui sistem ierarhic din cadrul unei organizații în care rădăcina corespunde directorului general, iar fiecare alt angajat are un singur manager - elaborarea unei aplicații cu arbori binari de căutare pentru gestionarea rapidă a comenzilor unei firme de livrare de produse, identificate prin chei unice	
CS 6.5. Implementează aplicații în limbaj de programare care integrează clase personalizate, pentru a modulariza și organiza eficient codul în cadrul soluțiilor informatice	
- implementarea unei aplicații de gestionare a bibliotecii școlare, cu clase pentru cărți, utilizatori și evidența împrumuturilor - implementarea unei aplicații de evidență a comenzilor într-un magazin online, cu clase pentru produse și depozit - implementarea unui simulator de joc pentru participarea la concursul de robotică, având clase pentru „Echipament”, „Robot” și „Elev”	

CONȚINUTURI ALE ÎNVĂȚĂRII

Clasa a XI-a

Domenii de conținut	Conținuturi
1. Organizarea conceptuală a datelor	1.1. Modelul conceptual rețea - graf - concepte de bază și caracteristici: graf neorientat și orientat, nod și vârf, muchie și arc, adiacență, incidență, grad, grad interior, grad exterior, lanț și drum, lanț elementar și drum elementar, lanț simplu și drum simplu, lanț compus și drum compus, ciclu și circuit, ciclu elementar și circuit elementar; - metode de reprezentare a unui graf: diagramă, matrice de adiacență, liste de adiacență, liste de succesori, liste de predecesori, liste de muchii, liste de arce; - concepte de bază și caracteristici ale unor grafuri asociate unui graf: subgraf, graf parțial; - concepte de bază, caracteristici și proprietăți ale unor tipuri de grafuri: graf complet, graf conex, graf tare conex (matricea drumurilor), graf ponderat (matricea costurilor), graf hamiltonian (lanț hamiltonian, ciclu hamiltonian), graf eulerian (lanț eulerian, ciclu eulerian); - repere pentru aplicarea algoritmilor de bază în prelucrarea grafurilor. 1.2. Model conceptual ierarhic - arbore - caracteristici și definiții echivalente ale unui arbore, respectiv arbore cu rădăcină; - concepte de bază într-un arbore cu rădăcină și caracteristici ale acestora: rădăcină, nivel al unui nod, nod frunză, nod descendent direct/fiu, nod descendent, nod ascendent direct/tată, nod ascendent, nod frate; - metode de reprezentare a unui arbore cu rădăcină: diagramă, referințe ascendente, referințe descendente; - caracteristici ale arborilor ordonați, arborilor binari (cu subtipul arbore binar de căutare) - metode de reprezentare a unui arbore binar cu referințe descendente (liste de fii); - repere pentru aplicarea algoritmilor de bază pentru prelucrarea arborilor. 1.3. Model conceptual obiectual – clase și obiecte - concepte avansate și caracteristici: clasă, obiect, moștenire, clasă de bază, clasă derivată; - repere pentru aplicarea algoritmilor de bază cu date organizate obiectual.
2. Strategii de rezolvare a problemelor	2.1. Metoda Backtracking - caracteristici ale metodei Backtracking; - repere pentru aplicarea metodei de generare sistematică a unor secvențe de valori. 2.2. Algoritmi specifici pentru prelucrarea grafurilor - caracteristici, etape ale metodelor de parcurgere în lățime (Breadth First Search - BFS), respectiv în adâncime (Depth First Search – DFS) și repere pentru aplicarea acestora; - repere pentru utilizarea metodelor de parcurgere pentru verificarea proprietății de conexitate a unui graf neorientat și pentru determinarea componentelor conexe; - repere pentru utilizarea metodelor de parcurgere pentru determinarea matricei drumurilor/lanțurilor unui graf; - repere pentru utilizarea matricei drumurilor pentru verificarea proprietății de tare conexitate și determinarea componentelor tare conexe ale unui graf orientat; - caracteristici, etape ale algoritmului lui Dijkstra pentru determinarea drumurilor de cost minim într-un graf ponderat și repere pentru aplicarea acestuia; - repere pentru adaptarea metodei de parcurgere în adâncime pentru determinarea unui ciclu eulerian; - caracteristici, etape ale algoritmului lui Prim pentru determinarea arborelui de acoperire de cost minim într-un graf ponderat și repere pentru aplicarea acestuia. 2.3. Algoritmi specifici pentru prelucrarea arborilor - caracteristici, etape ale metodelor de parcurgere a nodurilor unui arbore binar în preordine, inordine și postordine și repere pentru aplicarea acestora; - caracteristici, etape ale metodelor de căutare și de inserare a unei chei într-un arbore binar de căutare și repere pentru aplicarea acestora.
3. Memorarea datelor și organizarea codului în limbaj de programare	3.1. Programare orientată pe obiecte - sintaxa definirii unei clase, niveluri de acces la membrii clasei, constructor, destructor/finalizer, clasă derivată în Python;

Domenii de conținut	Conținuturi
	- repere pentru definirea unor clase simple și a claselor derivate. 3.2. Paradigme de bază în programare - caracteristici, tipuri de paradigme (de exemplu, procedurală, obiectuală, declarativă); - concepte, funcționalități, avantaje și dezavantaje ale altor limbaje de programare (de exemplu, C, C++, Java, JavaScript, C#, J#).

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 măsoară? 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.

Un context de învățare poate viza simultan mai multe competențe, mai ales atunci când acestea sunt interconectate. De exemplu, pentru rezolvarea unei probleme cu ajutorul calculatorului se vizează, în cadrul aceleiași activități de învățare, atât competențe de nivel cognitiv scăzut (identificarea principalelor caracteristici ale modelelor conceptuale și operaționale implicate, explicarea principiilor care stau la baza acestora, în contextul particular al problemei, utilizarea lor în rezolvare) dar și competențe de nivel cognitiv ridicat (analizarea caracteristicilor modelelor conceptuale și operaționale specifice problemei, evaluarea corectitudinii și eficienței soluțiilor propuse și elaborarea algoritmului personalizat).

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, algoritimizarea, 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 algoritmi 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 algoritmi 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 metode/algoritmi 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, urmărirea 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 XI-a, specializarea matematică-informatică:

1. *Strategii de rezolvare a problemelor. Metoda Backtracking*
2. *Organizarea conceptuală a datelor. Model conceptual rețea – graf*
3. *Strategii de rezolvare a problemelor. Algoritmi specifici pentru prelucrarea grafurilor*
4. *Organizarea conceptuală a datelor. Model conceptual ierarhic – arbore*
5. *Strategii de rezolvare a problemelor. Algoritmi specifici pentru prelucrarea arborilor*
6. *Organizarea conceptuală a datelor. Model conceptual obiectual*
7. *Memorarea datelor și organizarea codului în limbaj de programare. Programare orientată pe obiecte*
8. *Memorarea datelor și organizarea codului în limbaj de programare. Paradigme de bază în programare*

Pentru predarea metodei Backtracking se recomandă precizarea unor repere avute în vedere pentru rezolvarea problemelor (de exemplu, ce este o soluție dintre cele cerute, ce reprezintă un element al acestora, care sunt condițiile de validare a unei valori propuse, care este condiția de obținere a unei soluții complete) și exemplificarea acestora pentru cazuri concrete. Se recomandă abordarea unor rezolvări cu aplicarea generării de elemente combinatoriale (de exemplu, permutări, aranjamente, combinații, produs cartezian), eventual cu restricții privind relațiile dintre elemente, probleme clasice în care o soluție se poate reprezenta ca o succesiune de perechi, în care poziția are o anumită semnificație (de exemplu, problema reginelor, în care o soluție este o succesiune de poziții pe tabla de șah, perechi linie-coloană, iar numărul de ordine în cadrul soluției reprezintă linia și valoarea corespunzătoare reprezintă coloana, problema colorării unei hărți, în care o soluție este o succesiune de perechi țară-culoare, iar numărul de ordine în cadrul soluției reprezintă țara și valoarea corespunzătoare reprezintă culoarea), probleme în care sunt preluate valori parțiale calculate la pașii anteriori (de exemplu, problema partițiilor unui număr, problema plății unei sume cu anumite tipuri de monede, problema partițiilor unei mulțimi), probleme de optim (de exemplu, problema unui comis-voiajor) ca model, apoi realizarea unor probleme asemănătoare care presupun adaptări, reinterpretări, extinderi.

Se recomandă ca în cadrul temelor Model conceptual rețea – graf și Metode de prelucrare a grafurilor reprezentările grafurilor (diagramă, matrice de adiacență, liste) să fie introduse succint, cu exemple intuitive (hartă, rețea de prieteni), iar accentul să fie pus ulterior pe utilizarea algoritmilor specifici, în mod structurat, urmând etape corespunzătoare pentru elaborarea programelor. Programa este flexibilă din punctul de vedere a ordinii de predare, astfel încât se pot preda, de exemplu, în paralel grafurile neorientate și orientate, apoi arborii, pentru a elimina redundanțele și a pune în evidență diferențele dintre acestea, sau se pot preda succesiv, pentru a fixa noțiunile specifice, de exemplu, grafuri neorientate, grafuri orientate, arbori sau grafuri neorientate, arbori, grafuri orientate. Astfel, pentru grafuri, se pot preda concepte de bază (definiții ale grafurilor neorientate/orientate, concepte ca adiacență, grade, lanț/drum, ciclu/circuit), metode de reprezentare și aplicații cu algoritmi de bază, de exemplu, pentru determinarea gradului unui nod/vârf, a nodurilor izolate, verificarea proprietății de lanț/drum pentru un șir de valori. După aceea, se pot prezenta metodele de parcurgere a grafurilor, grafuri asociate și tipuri de grafuri: grafuri complete, grafuri conexe (definiții și utilizarea metodelor de parcurgere în aplicații), grafuri tare conexe (definiții, algoritmi specializați pentru determinarea matricei drumurilor și utilizarea lor în aplicații), grafuri ponderate (definiții, algoritmi specializați pentru determinarea unor lanțuri/drumuri de cost minim și utilizarea acestora în aplicații), grafuri hamiltoniene și euleriene (definiții și teoreme, algoritmi specifici). La liceu nu se studiază grafuri cu bucle, grafuri infinite sau multigrafuri, considerându-se că oricare arc/muchie are extremități distincte și oricare două arce/muchii diferă prin cel puțin una dintre extremități. De remarcat că în lucrările de specialitate, algoritmul Roy-Floyd se regăsește și cu denumirea Floyd-Warshall.

În cadrul temei Model conceptual ierarhic – arbore, se recomandă să se pornească de la modele vizuale reale (arbore genealogic, ierarhie școlară, meniuri de site), iar după aceea să fie introduse concepte formale. Implementarea arborilor binari în Python se face utilizând clasa list, fiecare înlanțuire cu un nod ascendent/descendent fiind dată prin poziția la care se află acesta.

Pentru tema Programare orientată pe obiecte, se recomandă evidențierea faptului că programarea orientată pe obiecte nu înseamnă doar „alt tip de sintaxă”, ci un mod de gândire în care problemele se modelează folosind entități (obiecte) și tipuri (clase), iar accentul se mută de la „ce face programul pas cu pas” la „cine (obiectul) face acțiunea”. Moștenirea evidențiază „specializarea” claselor, cu preluarea membrilor care definesc comportamente generale și aplicarea principiului polimorfismului pentru adaptarea acestora la specificul claselor derivate.

Proiectele interdisciplinare și învățarea pe baza unor aplicații care să conecteze teoria cu lumea reală (de exemplu, pentru grafuri, modelarea rețelelor sociale, a rutelor între orașe, pentru arbori, modelarea unor ierarhii organizaționale, a arborelui genealogic) 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.

Pentru tema Paradigme de bază în programare se vor prezenta succint conținuturile din programă, având în vedere diferențe și asemănări, prezentarea unui exemplu de cod în Python și în alte limbaje pentru principalele aspecte evidențiate, punând în valoare facilitățile limbajelor/paradigmelor.

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,

Nume și prenume	Grad didactic/Titlu științific	Instituție de apartenență, localitate, județ
		Facultatea de Automatică și Calculatoare
NIȚU Alina	profesor, gradul didactic I	Liceul Teoretic „Ovidius”, Constanța, județul Constanța
NURLA Aidan	profesor, gradul didactic I	Colegiul Național Militar „Alexandru Ioan Cuza”, Constanța, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forțelor 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 Național Militar „Dimitrie Cantemir”, Breaza, județul Prahova
TEGLAȘ Maria	profesor, gradul didactic II	Colegiul Național 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

Anexa nr. 4

Programa școlară
pentru disciplina

Informatică

Clasa a XII-a

Curriculum de specialitate (CS)

Filiera teoretică, profilul real, specializarea matematică-informatică
Filiera vocațională, profilul militar,
specializarea matematică-informatică militară

Învățământ liceal

- 2026 –

Anexa nr. 4**Programa școlară pentru disciplina *Informatică*, clasa a XII-a, curriculum de specialitate (CS)**

- *Filiera teoretică, profilul real, specializarea matematică-informatică*
- *Filiera vocațională, profilul militar, specializarea matematică-informatică militară*

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 matematică-informatică, alocarea orară este:

- clasa a IX-a – 2 ore/săptămână, dintre care o oră pentru studiu teoretic și o oră pentru activități practice;
- clasa a X-a – 2 ore/săptămână, dintre care o oră pentru studiu teoretic și o oră pentru activități practice;
- clasa a XI-a – 4 ore/săptămână, dintre care două ore pentru studiu teoretic și două ore pentru activități practice;
- clasa a XII-a – 3 ore/săptămână, dintre care o oră pentru studiu teoretic și două ore pentru 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);
- Cadrul de competențe de inteligență artificială pentru elevi (AI competency framework for students), UNESCO 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 limbajelor de programare Python și 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 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. 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.

Programa școlară pentru disciplina Informatică, clasa a XII-a, curriculum de specialitate (CS);

filiera teoretică, profilul real, specializarea matematică-informatică; filiera vocațională, profilul militar, specializarea matematică-informatică militară

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 experiența 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 XII-a
CS 1.1. Identifică principalele caracteristici ale organizării datelor în cadrul unor modele conceptuale avansate - pentru proiectarea bazelor de date sau învățare automată, pentru a structura și accesa date în vederea prelucrării acestora
- identificarea elementelor principale ale unui model entitate-relație (entitate, atribut, relație, identificator unic, tipuri de identificatori unici - natural/artificial, simplu/compus) în contextul gestionării informațiilor despre cărți, elevi și împrumuturile realizate într-o bibliotecă școlară - recunoașterea tipurilor de cardinalitate (1:1, 1:M, M:M) dintr-o listă de exemple de relații între entități pentru gestionarea unui magazin online - recunoașterea etapelor de preprocesare a datelor (curățare, normalizare, codificare) într-un flux de analiză a datelor provenite din senzori medicali - identificarea conceptelor de bază (set de antrenare, set de testare, caracteristici, etichetă) într-un proiect practic ce presupune clasificarea automată a e-mailurilor în unele de tip spam/non-spam
CS 1.2. Identifică specificul, caracteristicile și etapele unor strategii de normalizare a modelului conceptual al unei probleme de gestiune
- enumerarea caracteristicilor datelor care corespund primei forme normale (FN1), în contextul unei baze de date pentru o bibliotecă, unde unele cărți au mai mulți autori - enumerarea caracteristicilor datelor care corespund celei de a doua forme normale (FN2), în contextul unei baze de date pentru o bancă, unde unele tabele au identificator compus - enumerarea caracteristicilor datelor care corespund celei de a treia forme normale (FN3), în contextul unei baze de date pentru o școală, unde pentru fiecare elev sunt memorate datele părinților
CS 1.3. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în învățarea automată
- enumerarea caracteristicilor învățării supervizate în contextul aplicării pentru soluționarea unor probleme practice pentru care avem acces la date etichetate de experți - enumerarea caracteristicilor învățării nesupervizate în contextul aplicării pentru soluționarea unor probleme practice pentru care nu avem acces la etichete - recunoașterea termenilor de bază (de exemplu, învățare supervizată, învățare nesupervizată, clasificare, regresie, clusterizare) în contextul unor situații practice din viața reală - identificarea caracteristicilor algoritmului K-means într-un context de clasificare a clienților după comportamentul de cumpărare
CS 1.4. Identifică principalele comenzi SQL și ale limbajului de programare pentru a identifica soluția adecvată prelucrării datelor organizate în baze de date, în vederea rezolvării eficiente a problemelor informatice de gestiune
- identificarea tipurilor de comenzi SQL pentru gestionarea unei baze de date a unei biblioteci, prin clasificarea fiecărei comenzi în categoria corespunzătoare - descrierea structurii de bază a unei comenzi SELECT pentru interogarea unei baze de date a unui magazin online, prin enumerarea componentelor acesteia - recunoașterea comenzilor SQL pentru gestionarea bazei de date a unei clinici medicale, prin acțiuni corespunzătoare asupra tabelelor - enumerarea claselor și metodelor principale ale limbajului Python utilizate pentru conectarea la o bază de date pentru o aplicație de evidență a produselor - identificarea părților componente ale unui cod simplu de conectare la o bază de date (nume al clasei, metodă adecvată, parametri) pentru un script Python care accesează baza de date a unei clinici medicale - recunoașterea rolului bibliotecilor uzuale (sqlite3, mysql.connector) într-un fragment de cod Python pentru două aplicații diferite
CS 1.5. Identifică principalele elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în învățarea automată, în vederea rezolvării eficiente a problemelor informatice
- enumerarea bibliotecilor Python utilizate în învățarea automată (Pandas, Matplotlib, Scikit-learn) și precizarea scopului fiecăreia pentru înțelegerea ecosistemului de lucru cu date, specificând că Pandas gestionează date tabulare, Matplotlib vizualizări grafice și Scikit-learn algoritmi de machine learning

Clasa a XII-a
- identificarea claselor principale (<i>DataFrame</i>, <i>Series</i>, <i>LinearRegression</i>) în fragmente de cod Python pentru recunoașterea structurilor de date într-un script de analiză a vânzărilor, distingând <i>DataFrame</i> ca tabel complet, <i>Series</i> ca o coloană și <i>LinearRegression</i> ca model de regresie liniară - recunoașterea tipurilor de grafice (linie, bare, histogramă, box-plot, scatter) generate de <i>Matplotlib</i> pentru interpretarea corectă a vizualizărilor dintr-un raport de analiză, identificând graficul de linie pentru evoluții temporale, bare pentru comparații categoricale, histogramă pentru distribuții, box-plot pentru statistici descriptive și scatter pentru corelații

CG 2 - Explică principiile 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 XII-a
CS 2.1. Explică principiile de organizare a datelor în cadrul unor modele conceptuale avansate - pentru proiectarea bazelor de date sau învățare automată, pentru a le gestiona eficient
- explicarea diferenței dintre o entitate și o instanță a unei entități prin exemplul gestionării informațiilor despre elevi, profesori și discipline într-o școală - explicarea modului în care o relație M:M se transformă în două relații 1:M, de regulă barate, introducând o entitate de intersecție, prin exemplul unui sistem de notare a elevilor la fiecare disciplină, de-a lungul unui an de studiu - interpretarea unei diagrame simple entitate-relație și explicarea sensului simbolurilor utilizate (bare, opționalități, cardinalități) pentru un sistem de gestionare a comenzilor într-un restaurant, unde se analizează relațiile dintre clienți, comenzi și produse - explicarea diferenței dintre date brute și date preprocesate folosind ca exemplu un set de date despre pacienți (unde unele valori lipsesc sau sunt exprimate în unități de măsură diferite) - ilustrarea prin exemple a modului în care structura datelor influențează performanța modelului într-un caz practic de predicție a prețului apartamentelor - explicarea, pe baza unui exemplu din domeniul sănătății, a modului în care datele incomplete sau neomogene pot afecta performanța unui model de diagnostic automat
CS 2.2. Explică etapele unor strategii de normalizare a modelului conceptual al unei probleme de gestiune
- explicarea importanței normalizării unei baze de date pentru un sistem de gestionare a comenzilor unui magazin online, prin evidențierea reducerii redundanțelor, a prevenirii inconsistențelor și a simplificării prelucrării datelor - explicarea diferențelor dintre prima, a doua și a treia formă normală, pentru un sistem de gestionare a comenzilor unui magazin online - interpretarea unui model conceptual dat pentru un sistem de gestionare a vânzărilor unui magazin, prin evidențierea respectării formelor normale
CS 2.3. Explică etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în învățarea automată
- explicarea principiilor învățării nesupervizate și clusterizării pe baza unui exemplu de grupare a utilizatorilor unei platforme video în funcție de preferințe - explicarea diferențelor dintre clasificare și regresie pe baza unui set de date cu vânzări lunare (regresie) versus tipul clientului (clasificare) - ilustrarea modului în care algoritmul KNN utilizează vecinii apropiați pentru a clasifica un element într-un scenariu simplu de clasificare a florilor (setul Iris) - explicarea modului în care alegerea atributelor poate afecta performanțele unui algoritm de învățare automată utilizat pentru rezolvarea unei probleme de predicție a prețului unui apartament
CS 2.4. Explică rolul componentelor principalelor comenzi SQL și ale limbajului de programare pentru a identifica soluția adecvată prelucrării datelor organizate în baze de date, în vederea rezolvării eficiente a problemelor informatice de gestiune
- explicarea diferenței dintre comenzile DDL și DML pentru gestionarea bazei de date a unei școli, prin evidențierea acțiunilor corespunzătoare asupra structurii, respectiv asupra conținutului tabelor - explicarea utilizării funcțiilor agregate într-o interogare pentru analiza vânzărilor dintr-un supermarket, prin rezultatul obținut în urma aplicării lor - interpretarea unui exemplu de interogare SQL și explicarea rezultatului pe baza datelor afișate pentru o bază de date cu angajați și departamente, prin analiza pas cu pas a fiecărei clauze - explicarea, în propriile cuvinte, a diferenței dintre obiectul de tip <i>Connection</i> și cel de tip <i>Cursor</i> pentru înțelegerea arhitecturii de lucru cu baze de date în Python, evidențiind faptul că obiectul de tip <i>Connection</i> reprezintă legătura fizică la baza de date (sesiunea de comunicare), iar obiectul de tip <i>Cursor</i> este cel care execută comenzile SQL și gestionează rezultatele, ilustrând analogia cu o carte (<i>Connection</i>) și un semn de carte (<i>Cursor</i>) care marchează poziția curentă - descrierea pașilor necesari pentru realizarea unei conexiuni la o bază de date și executarea unei interogări pentru un script Python care accesează o bază de date SQLite cu informații despre produse, detaliind secvența: importarea bibliotecii (<i>sqlite3</i>), crearea conexiunii cu <i>connect()</i>, crearea obiectului <i>Cursor</i> cu <i>cursor()</i>, executarea interogării cu <i>execute()</i>, preluarea rezultatelor cu <i>fetchall()</i> sau <i>fetchone()</i>, și închiderea resurselor cu <i>close()</i>

Clasa a XII-a
- interpretarea unui fragment de cod Python care utilizează obiecte de tip <i>Connection</i> și <i>Cursor</i>, prin explicarea secvenței de operații efectuate pentru a extrage date din tabela <i>Comenzi</i>, identificând fiecare linie de cod, rolul acestora (conectare, executare, preluare date, închidere), ordinea de executare și modul în care datele sunt transferate de la baza de date în vederea prelucrării lor
CS 2.5. Explică rolul principalelor elemente ale limbajului de programare utilizate pentru prelucrarea datelor în învățarea automată, în vederea rezolvării eficiente a problemelor informatice
- explicarea modului în care un obiect <i>DataFrame</i> reprezintă datele sub formă de tabel pentru înțelegerea structurii de date <i>Pandas</i>, prin organizarea informațiilor - explicarea modului în care <i>DataFrame</i>-urile <i>Pandas</i> stochează și manipulează datele pentru alegerea structurii potrivite în analiza datelor - interpretarea unui fragment de cod care construiește o regresie liniară simplă cu <i>Scikit-learn</i> pentru predicția prețurilor unor apartamente pe baza suprafeței, prin antrenarea unui model pe datele disponibile

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

Clasa a XII-a
CS 3.1. Utilizează modul de organizare și prelucrare a datelor în cadrul unor modele conceptuale avansate – la proiectarea bazelor de date sau la învățare automată, pentru a rezolva probleme de gestionare a datelor
- aplicarea regulilor pentru transformarea unei relații <i>M:M</i> prin exemplul unui sistem de închiriere filme, unde se utilizează regula introducerii unei entități de intersecție, pentru împrumuturi - completarea unei diagrame entitate-relație parțiale cu marcatorii adecvați pentru atribute, respectând regulile de identificare unică pentru un sistem de gestiune a vânzărilor, care conține entitățile <i>PRODUS</i>, <i>CLIENT</i> și <i>COMANDĂ</i> - aplicarea unor operații de augmentare a datelor (de exemplu, rotirea sau scalarea imaginilor) pentru un set de imagini cu fructe - aplicarea operației de normalizare a valorilor numerice ale unui set de date cu puține exemple provenit dintr-o bază de date cu indicatori economici, manual, prin calculul valorilor normalizate pe baza valorilor minime și maxime, pentru a aduce toate valorile pe aceeași scară - aplicarea unor criterii pentru separarea unui set de date despre rezultate școlare în seturi de antrenare și testare, în vederea asigurării existenței unor date noi care să fie utilizate pentru evaluarea unui algoritm de învățare automată
CS 3.2. Aplică strategii de normalizare a modelului conceptual al unei probleme de gestiune
- aplicarea strategiei pentru obținerea primei forme normale (<i>FN1</i>) pentru un sistem de gestionare a comenzilor unui magazin online, prin tratarea adecvată a fiecărui atribut al unei entități care nu are un caracter atomic, fiind format din mai multe componente care sunt utilizate independent - aplicarea strategiei pentru obținerea celei de a doua forme normale (<i>FN2</i>) pentru un model conceptual al unei baze de date, prin tratarea adecvată a dependențelor unor atribute de componente ale identificatorului unic compus, pentru un sistem de gestiune a activităților nonformale dintr-o școală - aplicarea strategiei pentru obținerea celei de a treia forme normale (<i>FN3</i>) a unui model conceptual al unei baze de date, prin tratarea adecvată a dependențelor unor atribute de alte atribute diferite de identificatorul unic, cu respectarea integrității datelor pentru un sistem de gestiune a angajaților și departamentelor unei companii
CS 3.3. Aplică algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în învățarea automată
- aplicarea algoritmului <i>K-means</i> pe un set de date cu coordonate 2D reprezentând sediile a 15 magazine dintr-un oraș în vederea identificării a trei zone optime pentru amplasarea depozitelor de distribuție, urmărind cum centroizii se deplasează progresiv către pozițiile finale și cum magazinele sunt reatribuite la fiecare iterație - aplicarea algoritmului de regresie liniară pentru soluționarea unei probleme de predicție a prețurilor caselor pornind de la un set de date ce conține cinci proprietăți pentru care cunoaștem suprafața și prețul - aplicarea algoritmului de clasificare <i>KNN</i> pentru un set de date simplu cu fructe (mere, portocale, banane) descrise prin greutate și diametru
CS 3.4. Utilizează principalele comenzi SQL și ale limbajului de programare pentru a identifica soluția adecvată prelucrării datelor organizate în baze de date, în vederea rezolvării eficiente a problemelor informatice de gestiune
- aplicarea unei interogări <i>SQL</i> pentru baza de date a unui liceu, pentru afișarea elevilor cu media peste 9, în ordine alfabetică - aplicarea unei comenzi <i>JOIN</i> pentru obținerea unui raport școlar, prin extragerea informațiilor din două tabele relaționate - aplicarea unei funcții de conversie într-o interogare pentru a transforma un tip de date numeric în șir de caractere prin generarea de etichete cu prețuri într-un magazin - utilizarea instrumentelor limbajului Python pentru conectarea la o bază de date <i>SQLite</i> și afișarea conținutului unei tabele pentru vizualizarea produselor dintr-un magazin, prin crearea conexiunii, a cursorului, executarea interogării și afișarea rezultatelor - utilizarea metodelor <i>execute()</i> și <i>fetchall()</i> și a comenzii <i>SELECT</i> pentru preluarea și afișarea rezultatelor unei interogări în vederea extragerii tuturor comenzilor dintr-o perioadă specificată

Clasa a XII-a
- utilizarea metodelor <code>execute()</code>, <code>close()</code> și <code>commit()</code> și a comenzii <code>INSERT INTO</code>, pentru a insera date într-o tabelă, folosind conexiunea și cursorul corespunzător și închizând conexiunea corect, în contextul adăugării unui client nou în tabela <i>Cienti</i>
CS 3.5. Utilizează principalele elemente ale limbajului de programare pentru prelucrarea datelor în învățarea automată, în vederea rezolvării eficiente a problemelor informatice
- utilizarea metodelor din <i>Pandas</i> pentru citirea unui set de date dintr-un fișier <i>Note.csv</i> și afișarea valorilor statistice de bază (medie, minim, maxim) în contextul analizei situației școlare - aplicarea metodelor din <i>Matplotlib</i> pentru reprezentarea grafică a datelor privind vânzările lunare ale unui magazin, prin realizarea unor grafice adecvate (de exemplu, grafic liniar, diagramă cu bare) - utilizarea metodelor <code>fit()</code> și <code>predict()</code> ale clasei <i>LinearRegression</i> din <i>Scikit-learn</i> pentru a antrena un model de regresie liniară și realizarea unei predicții pentru consumul de combustibil pe baza greutății mașinii și a distanței parcurse

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 XII-a
CS 4.1. Analizează caracteristicile, modul de prelucrare, avantajele și dezavantajele utilizării unor modele conceptuale avansate – pentru proiectarea bazelor de date sau pentru învățare automată, pentru a alege structura adecvată în rezolvarea unor probleme concrete
- analizarea unui model conceptual dat, corespunzător activității de programare a cursanților dintr-o școală de conducători auto, pentru identificarea corespondenței dintre scenariu și entitățile și relațiile modelului - analizarea unui model entitate-relație existent pentru a identifica erorile de cardinalitate pentru un sistem de gestionare a rezervărilor la un hotel, prin examinarea relațiilor dintre clienți, camere și rezervări - analizarea unui model entitate-relație dat pentru a identifica erorile de redundanță pentru un sistem de gestionare a comenzilor într-un magazin online - analizarea structurii unui set de date real (număr de atribute, tipuri de variabile, valori lipsă), pentru identificarea pașilor de preprocesare necesari - compararea diferitelor metode de transformare a datelor (scalare min-max vs. standardizare), pentru identificarea contextului potrivit pentru fiecare, pornind de la o problemă din viața reală - analizarea corelațiilor dintre variabile într-un set de date despre obiceiuri alimentare și starea de sănătate, pentru a identifica atributele relevante pentru predicția finală
CS 4.2. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de normalizare a modelului conceptual al unei probleme de gestiune
- analizează avantajele unui model conceptual ce respectă prima formă normală (FN1), comparativ cu un model care nu respectă această formă, în contextul obținerii orașului de domiciliu al unei persoane, pentru cazul în care acesta este inclus în adresă (utilizarea unei părți a unui atribut) și pentru cazul în care el reprezintă un câmp individual (utilizarea acestuia ca o valoare atomică) - detectarea dependențelor funcționale dintre atribute și clasificarea lor pentru un sistem de gestiune a angajaților și proiectelor dintr-o companie, prin identificarea dependențelor parțiale și a celor tranzitive - compararea a două modele conceptuale ale aceleiași situații prin evidențierea celei care respectă principiile normalizării pentru un sistem de gestionare a comenzilor și clienților unui magazin de electronice
CS 4.3. Analizează comportamentul, aplicabilitatea, avantajele și dezavantajele utilizării unor algoritmi specializați pe clase de probleme pentru învățarea automată
- analizarea diferențelor dintre învățarea supervizată și cea nesupervizată, prin completarea unui tabel comparativ cu exemple concrete (de exemplu, predicția notelor vs. gruparea elevilor după interese) - compararea performanței algoritmilor <i>K-means</i>, regresie liniară și <i>KNN</i>, prin aplicarea fiecăruia pe seturi de date adecvate problemelor pe care le pot rezolva, din perspectiva metricii specifice fiecărui tip de problemă (de exemplu, coeficientul <i>Silhouette</i> pentru <i>K-means</i>, eroarea medie pătratică pentru regresie liniară, acuratețea sau matricea de confuzie pentru <i>KNN</i>) - interpretarea grafică a rezultatelor obținute de diferiți algoritmi, identificând modelele de date și abaterile față de tendință - analizarea unui caz (de exemplu, recomandarea de filme) pentru a identifica tipul de algoritm de învățare automată cel mai potrivit și justificarea alegerii în funcție de natura problemei și a datelor disponibile
CS 4.4. Analizează aplicabilitatea, avantajele și dezavantajele utilizării comenzilor SQL și a elementelor specifice limbajului de programare, în vederea rezolvării eficiente a problemelor informatice de gestiune
- examinarea unei comenzi SQL pentru o interogare care extrage comenzile dintr-un sistem de gestiune a unui magazin online, prin identificarea eventualelor erori de sintaxă sau de logică - compararea a două interogări care obțin același rezultat, dar folosesc metode diferite pentru afișarea produselor comandate de un client specific dintr-un sistem e-commerce, prin evaluarea avantajelor și dezavantajelor fiecărei abordări - examinarea părților componente ale unei interogări complexe pentru un raport de vânzări lunar, prin explicarea rolului fiecăreia

Clasa a XII-a	
- analizarea unui cod Python care realizează o conexiune la o bază de date și identificarea posibilelor erori (lipsa commit(), conexiune neînchisă) pentru un script care actualizează prețurile produselor, verificând dacă lipsește commit() după UPDATE (modificările nu se salvează), dacă conexiunea rămâne deschisă (resurse blocate) sau dacă nu există tratare de excepții - compararea a două modalități de executare în Python a comenzilor SQL (prin execute() și prin executemany()) și determinarea cazului potrivit fiecăreia pentru inserarea de date în tabela Elevi, stabilind că execute() este eficient pentru o singură înregistrare, iar executemany() este optim pentru inserarea mai multor înregistrări dintr-o listă, reducând numărul de apeluri - descompunerea unui exemplu complet (creare, inserare, interogare, închidere) pentru a evidenția ordinea corectă a apelurilor de metode pentru un script (de exemplu, pentru gestiunea unei biblioteci), secvențializând: connect() → cursor() → execute(CREATE TABLE) → execute(INSERT) → commit() → execute(SELECT) → fetchall() → close(), identificând dependențele dintre pași	
CS 4.5. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor elemente ale limbajului de programare, pentru a identifica soluția adecvată prelucrării datelor în învățarea automată, în vederea rezolvării eficiente a problemelor informatice	
- analizarea unui program Python care citește, curăță și vizualizează datele pentru a identifica ordinea corectă a pașilor de prelucrare pentru un script ce analizează tranzacțiile comerciale - compararea rezultatelor obținute cu funcția describe() din Pandas cu valorile calculate manual pe baza operațiilor de bază cu liste pentru statisticile de bază (medie, minim, maxim) pe un set de date cu temperaturi, verificând concordanța calculului - analizarea unui grafic complex pentru a explica modul în care datele sunt combinate și afișate în Matplotlib pentru un panou de control cu grafice multiple despre performanța elevilor	

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

Clasa a XII-a	
CS 5.1. Argumentează alegerile făcute în cadrul proiectării unor modele conceptuale avansate - pentru proiectarea bazelor de date sau pentru învățare automată, în vederea rezolvării unor probleme informatice concrete	
- evaluarea unui model entitate-relație din perspectiva respectării regulilor de identificare unică a instanțelor pentru un sistem de gestionare a închirierilor de mașini - evaluarea unui model entitate-relație din punctul de vedere al rezolvării adecvate a relațiilor M:M, pentru un sistem de gestionare a unei biblioteci digitale - justificarea deciziei de a folosi o relație ierarhică în locul uneia recursive, pentru un sistem de gestionare a structurii organizaționale a unei companii, prin analiza relației dintre angajați și manageri - evaluarea calității unui set de date privind tranzacțiile bancare (din punctul de vedere al completitudinii și relevanței) înainte de utilizarea într-un algoritm de învățare automată - evaluarea impactului unei variante de normalizare asupra preciziei unui model de învățare automată aplicat pe un set de date despre diabet - argumentarea deciziei de a folosi codificarea de tip one-hot în locul asocierii unei etichete pentru variabile categorice ce descriu tipul unei mașini (sedan, SUV), în funcție de algoritmul de învățare automată utilizat	
CS 5.2. Evaluează corectitudinea și eficiența modelelor conceptuale ale unei probleme de gestiune din punctul de vedere al normalizării	
- evaluarea corectitudinii procesului de normalizare în cadrul bazelor de date aplicat pentru un sistem de gestiune al comenzilor și clienților unui magazin - argumentarea avantajelor și dezavantajelor utilizării formelor normale pentru un sistem de gestionare a comenzilor și stocurilor unui depozit, prin evidențierea avantajelor precum reducerea redundanțelor și creșterea consistenței datelor și a dezavantajelor precum creșterea numărului de tabele și complexitatea interogărilor - justificarea păstrării sau eliminării unei relații dintr-un model ERD în funcție de impactul asupra integrității și redundanței datelor pentru un sistem de gestionare a comenzilor și produselor unui magazin - identificarea atributelor care provoacă denormalizarea într-o bază de date pentru gestionarea vânzărilor dintr-un magazin, prin evidențierea atributelor care se repetă sau depind de alte atribute decât cheia primară	
CS 5.3. Evaluează corectitudinea și eficiența soluțiilor cu algoritmi specializați pe clase de probleme pentru învățare automată	
- evaluarea performanței unui model de regresie liniară pentru estimarea consumului de combustibil pe baza greutateii mașinii, prin compararea valorilor reale cu cele prezise și calcularea erorii medii (Mean Absolute Error sau Mean Squared Error) - argumentarea alegerii algoritmului potrivit (K-means, regresie liniară sau KNN) în funcție de scopul analizei și tipul datelor - formularea concluziilor privind precizia și utilitatea modelului KNN pentru clasificarea tipurilor de fructe pe baza caracteristicilor lor (greutate, culoare, diametru), folosind o valoare concretă a lui k (de exemplu, k=3), prin compararea valorilor reale și prezise obținute	

Clasa a XII-a
CS 5.4. Evaluează corectitudinea și eficiența secvențelor de cod sau aplicațiilor care utilizează principalele comenzi SQL și ale limbajului de programare pentru prelucrarea datelor organizate în baze de date, în vederea rezolvării eficiente a problemelor informatice de gestiune
- evaluarea eficienței unei interogări SQL pentru un raport al unui sistem bancar care afișează clienții cu tranzacții suspecte, prin analizarea performanței interogării actuale și propunerea de eventuale versiuni îmbunătățite - validarea respectării principiilor de integritate referențială pentru baza de date a unei universități, prin examinarea consecințelor ștergerii unor date asupra înregistrărilor dependente - justificarea alegerii unui anumit tip de funcție (agregată vs. scalară) pentru obținerea rezultatului dorit în contexte diferite dintr-un sistem de resurse umane prin calcularea salariului mediu pentru un departament versus formatarea numelui complet al angajatului - evaluarea eficienței unui program Python care lucrează cu baze de date, din punctul de vedere al timpului de executare și al gestionării resurselor pentru o aplicație de import masiv de tranzacții, analizând dacă executemany() este folosit corect, conexiunea se deschide/închide repetat ineficient și dacă fetchall() consumă prea multă memorie pentru seturi mari de date - argumentarea alegerii unei metode de conectare (sqlite3 local vs. mysql.connector extern) în funcție de scopul aplicației pentru proiectarea unui sistem de evidență, justificând sqlite3 pentru aplicații desktop single-user cu portabilitate (de exemplu, catalog personal) sau mysql.connector pentru aplicații multi-user cu acces concurent (de exemplu, sistem școlar) - justificarea utilizării blocurilor try-except-finally pentru gestionarea erorilor și închiderea sigură a conexiunilor pentru un modul de actualizare a datelor critice, demonstrând că try asigură execuția codului riscant, except prinde erori de conexiune/sintaxă SQL fără a opri programul, iar finally garantează close() chiar dacă apar excepții
CS 5.5. Evaluează corectitudinea și eficiența aplicațiilor care utilizează elemente specifice de limbaj de programare pentru prelucrarea datelor în învățarea automată, în vederea rezolvării eficiente a problemelor informatice
- evaluarea performanței unui model de regresie liniară pentru predicția prețurilor imobiliare, prin compararea valorilor reale și prezise și calcularea erorii medii pătratice (Mean Squared Error) folosind mean_squared_error din Scikit-learn, pentru setul de date de test - argumentarea alegerii tipului de reprezentare grafică (de exemplu, diagramă cu bare vs. histogramă) în funcție de tipul de date pentru vizualizarea relației notă-timp de studiu, pentru un raport privind notele și timpul de studiu dintr-o platformă de cursuri online - justificarea utilizării unei metode de curățare a datelor într-un set concret privind numărul de exerciții rezolvate de elevi pentru fiecare temă, cu valori incomplete corespunzătoare zilelor în care elevii au lipsit, prin evaluarea impactului asupra rezultatelor

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

Clasa a XII-a
CS 6.1. Proiectează reprezentări personalizate ale unor modele conceptuale avansate - pentru baze de date sau adecvate pentru învățare automată, în vederea rezolvării unor probleme
- proiectarea unui model entitate-relație personalizat pentru un sistem de gestionare a rezervărilor la un hotel, prin determinarea entităților și a relațiilor dintre acestea - proiectarea unui model entitate-relație personalizat pentru un sistem de gestionare a istoricului comenzilor și prețurilor într-un magazin online, pentru a asigura funcționarea corectă a bazei de date - îmbunătățirea unui model entitate-relație existent, cu justificarea modificărilor aduse pentru un sistem de gestionare a rezervărilor la un restaurant - schematizarea fluxului de prelucrare a datelor într-un proiect de învățare automată (colectare, curățare, transformare, antrenare) utilizat pentru detecția automată a unor probleme ale plămânilor, pornind de la imagini medicale de tip CT - proiectarea unui model conceptual al unui set de date (atribute, relații, etichete) pentru o problemă de predicție (de exemplu, performanța elevilor, vânzări, temperaturi) - crearea unui set de date sintetic care simulează valorile unor senzori de mediu (temperatură, umiditate, presiune) și care poate fi folosit pentru antrenarea unui model de predicție
CS 6.2. Proiectează soluții cu aplicarea personalizată a strategiilor pentru normalizarea modelului conceptual al unei probleme de gestiune
- proiectarea unui model conceptual pentru o aplicație de gestiune prin respectarea celor trei forme normale, pentru un magazin de electronice - elaborarea unui model conceptual normalizat pentru un sistem de gestionare a închirierilor de mașini, prin aplicarea fiecărei etape de normalizare - crearea unei reprezentări grafice (ERD) care demonstrează trecerea progresivă prin 1NF, 2NF și 3NF pentru un sistem de gestionare a rezervărilor la un hotel, prin prezentarea ERD inițial, apoi ERD după aplicarea fiecărei forme de normalizare
CS 6.3. Proiectează soluții cu aplicarea personalizată a algoritmilor specializați pe clase de probleme pentru învățare automată

Clasa a XII-a
- proiectarea unui mini-experiment de învățare automată, pornind de la un set de date colectat de elevi (de exemplu, obiceiuri de lectură, preferințe muzicale) - crearea unui model simplu de clasificare sau regresie utilizând aplicații educaționale (Orange Data Mining, Teachable Machine, Excel) - elaborarea unui scurt raport digital care descrie etapele aplicării algoritmului ales (pregătirea datelor, antrenarea modelului, interpretarea rezultatelor)
CS 6.4. Implementează secvențe de cod sau aplicații care integrează în mod personalizat comenzi SQL și elemente ale limbajului de programare pentru prelucrarea datelor organizate în baze de date, în vederea rezolvării eficiente a problemelor informatice de gestiune
- crearea tabelelor pentru un sistem de gestiune a unei bibliotecii, prin scrierea comenzilor DDL, respectând ordinea adecvată de definire a acestora - elaborarea unui set de comenzi SQL pentru analiza datelor unui sistem de rezervări hoteliere, prin generarea de rapoarte destinate managementului - crearea unui set de interogări cu JOIN, funcții agregat și condiții adecvate, pentru a genera un raport personalizat pentru un lanț de restaurante prin determinarea totalului de vânzări pentru câteva sucursale date - proiectarea unei aplicații Python care permite introducerea, modificarea și ștergerea datelor dintr-o bază de date (CRUD) pentru gestionarea inventarului unui magazin, implementând funcții separate <code>create_produc()</code>, <code>read_produce()</code>, <code>update_pret()</code> și <code>delete_produc()</code>, fiecare gestionând conexiunea, executând comanda SQL corespunzătoare (INSERT/SELECT/UPDATE/DELETE) și validând datele de intrare - elaborarea unui modul Python propriu care gestionează conexiunile și oferă funcții reutilizabile pentru execuția interogărilor pentru standardizarea accesului la baza de date în mai multe scripturi, creând clasa <code>DatabaseManager</code> cu metode <code>connect()</code>, <code>execute_query()</code>, <code>execute_update()</code> și <code>close()</code>, încapsulând logica de conexiune și tratare erori pentru reutilizare în diferite module - dezvoltarea unei aplicații complexe care folosește mai multe clase și metode (<code>Connection</code>, <code>Cursor</code>, <code>commit</code>, <code>rollback</code>) pentru operarea sigură a tranzacțiilor pentru un sistem bancar de transfer fonduri între conturi, implementând tranzacții atomice care debitează un cont și creditează altul, folosind <code>commit()</code> doar dacă ambele operații reușesc sau <code>rollback()</code> la eroare pentru menținerea consistenței datelor
CS 6.5. Implementează aplicații care integrează în mod personalizat elemente specifice de limbaj de programare pentru prelucrarea datelor în învățarea automată, în vederea rezolvării eficiente a problemelor informatice
- proiectarea unei aplicații Python complete care încarcă un fișier .csv, curăță datele, calculează statistici descriptive și afișează grafice multiple cu Matplotlib pentru analiza performanței sportive a elevilor dintr-un club sportiv, prin implementarea unui flux complet de activități - elaborarea unui mini-model de învățare automată pentru predicția mediei finale pe baza notelor la teste, utilizând clasa <code>LinearRegression</code> din Scikit-learn și împărțirea datelor în seturi de antrenare și testare prin funcția <code>train_test_split()</code> - crearea unui mini-proiect de învățare automată care include preprocesarea datelor, documentând fiecare etapă, pentru un set de date privind prețurile locuințelor, având în vedere curățarea și completarea valorilor lipsă, codificarea variabilelor categorice, normalizarea valorilor numerice și antrenarea unui model simplu, cu documentarea fiecărei etape într-un notebook Python

CONȚINUTURI ALE ÎNVĂȚĂRII

Clasa a XII-a

Domenii de conținut	Conținuturi
1. Organizarea conceptuală a datelor	1.1. Modelul conceptual relațional – entitate-relație - concepte de bază și caracteristici ale modelului conceptual entitate-relație: entitate (cu atribute opționale, obligatorii, volatile, respectiv cu identificatori unici, tipuri de identificatori unici – simplu/compus, natural/artificial), instanță, relație (opționalitate, cardinalitate, tipuri de relații din punctul de vedere al cardinalității, relații barate, relații ierarhice, relații recursive); - repere pentru identificarea entităților și reprezentarea convențională a diagramei entitate-relație (ERD), rezolvarea relațiilor M:M (mai mulți la mai mulți), reprezentarea istoricului datelor; - concepte de bază și caracteristici ale modelului fizic: tabelă (cu câmpuri, tipuri de date, cheie primară, cheie externe, înregistrări, constrângeri, reguli de integritate - de entitate și referențială); - repere pentru transformarea modelului conceptual în model fizic (mapare). 1.2. Modele conceptuale de organizare a datelor folosite în învățarea automată - concepte de bază și caracteristici ale învățării automate: date, atribute, etichete, model, antrenare (învățare), testare, predicție; - caracteristici ale seturilor de date utilizate în învățarea automată: tipuri de date, date categorice, date complete/lipsă, dimensiune și calitate a datelor; - repere pentru transformarea datelor în formate utilizabile de către algoritmi de inteligență artificială (normalizare, scalare, augmentare, codificare a datelor, tratare a datelor lipsă, organizare a datelor în fișiere sau tabele pentru prelucrare automată).
2. Strategii de rezolvare a problemelor	2.1. Forme normale pentru proiectarea modelului conceptual al unei probleme de gestiune - caracteristici și avantaje ale primelor trei forme normale (FN1, FN2, FN3) și repere pentru utilizarea acestora în proiectarea modelului conceptual al unei probleme de gestiune. 2.2. Algoritmi utilizați în învățare automată - concepte de bază și caracteristici: învățare nesupervizată, clusterizare; - caracteristici ale algoritmului K-means, pentru învățarea nesupervizată; - concepte de bază și caracteristici: învățare supervizată, regresie, clasificare; - caracteristici ale algoritmului de regresie liniară, pentru învățare supervizată; - caracteristici ale algoritmului KNN (k Nearest Neighbours), pentru învățare supervizată.
3. Memorarea datelor și organizarea codului în limbaj de programare	3.1. Comenzi SQL pentru gestionarea unei baze de date - caracteristici și sintaxă ale unor comenzi de bază pentru interogarea bazei de date (SQL): selecție, proiecție, ordonare, operatori, funcții scalare uzuale (de prelucrare a numerelor, a datelor calendaristice, a șirurilor de caractere, conversii), funcții agregate uzuale (pentru sumă, medie, numărare, minim, maxim, de tratare a valorii NULL), preluare a datelor din mai multe tabele (join); - caracteristici și sintaxă ale unor comenzi de bază pentru manipularea datelor (DML): inserare, modificare, ștergere; - caracteristici și sintaxă ale unor comenzi de bază pentru manipularea structurii bazei de date (DDL): creare a unei tabele, modificare a structurii unei tabele (adăugare a unui câmp, ștergere a unui câmp, modificare a unui câmp), ștergere a unei tabele; - repere pentru executarea comenzilor; - repere pentru identificarea și utilizarea altor comenzi, adecvate pentru prelucrarea bazelor de date, în funcție de nevoile proprii. 3.2. Clase ale limbajului Python pentru interacțiunea cu bazele de date - caracteristici; - metode ale claselor principale (de exemplu, Connection, Cursor) pentru operații uzuale: conectare la baza de date, executare a unei comenzi, închidere a conexiunii cu baza de date; - repere pentru identificarea și utilizarea altor clase și metode adecvate pentru interacțiunea cu bazele de date, în funcție de nevoile proprii. 3.3. Biblioteci ale limbajului Python pentru învățare automată

Domenii de conținut	Conținuturi
	- caracteristici, clase și metode specifice din biblioteca Matplotlib pentru vizualizarea datelor: reprezentarea grafică a datelor prin puncte, grafic liniar, diagramă cu bare, histogramă, diagramă box-plot; - caracteristici, clase și metode specifice din biblioteca Pandas pentru curățarea, manipularea și analiza statistică a datelor: tabele (DataFrames), eliminarea valorilor extreme, obținere a valorilor pentru un atribut, statistici descriptive (medie, deviație standard, minim, maxim); - caracteristici, clase și metode specifice din biblioteca Scikit-learn pentru rezolvarea problemelor de codificare și normalizare a datelor, clasificare, regresie și clusterizare: metode pentru antrenare, predicție și evaluare în rezolvarea problemelor de învățare automată; - repere pentru utilizarea algoritmilor prin metodele disponibile în biblioteci, pentru rezolvarea unor probleme de natură practică; - repere pentru identificarea și utilizarea altor clase și metode adecvate pentru învățare automată, 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 măsoară? 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.

Un context de învățare poate viza simultan mai multe competențe, mai ales atunci când acestea sunt interconectate. De exemplu, pentru rezolvarea unei probleme cu ajutorul calculatorului se vizează, în cadrul aceleiași activități de învățare, atât competențe de nivel cognitiv scăzut (identificarea principalelor caracteristici ale modelelor conceptuale și operaționale implicate, explicarea principiilor care stau la baza acestora, în contextul particular al problemei, utilizarea lor în rezolvare) dar și competențe de nivel cognitiv ridicat (analizarea caracteristicilor modelelor conceptuale și operaționale specifice problemei, evaluarea corectitudinii și eficienței soluțiilor propuse și elaborarea algoritmului personalizat).

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, algoritimizarea, 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 metode/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, urmărirea 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 XII-a, specializarea matematică-informatică și specializarea matematică-informatică militară

1. *Organizarea conceptuală a datelor. Model conceptual relațional – entitate-relație*
2. *Strategii de rezolvare a problemelor. Forme normale pentru proiectarea modelului conceptual al unei probleme de gestiune*
3. *Memorarea datelor și organizarea codului în limbaj de programare. Clase ale limbajului Python pentru interacțiunea cu bazele de date*
4. *Memorarea datelor și organizarea codului în limbaj de programare. Comenzi SQL pentru gestionarea unei bazei de date*
5. *Organizarea conceptuală a datelor. Modele conceptuale de organizare a datelor folosite în învățarea automată*
6. *Strategii de rezolvare a problemelor. Algoritmi utilizați în învățare automată*
7. *Memorarea datelor și organizarea codului în limbaj de programare. Biblioteci ale limbajului Python pentru învățare automată*

Se recomandă introducerea în tematica modelului entitate-relație prin exemple din viața reală (de exemplu, bibliotecă, magazin, școală) prin conectarea permanentă a conceptelor teoretice cu scenarii concrete, astfel încât elevii să înțeleagă conceptele specifice ca „entitate”, „relație”, „identificator unic”.

Procesul de învățare trebuie să fie centrat pe investigare și rezolvare de probleme, iar activitățile să favorizeze explorarea comparativă a soluțiilor. De exemplu, se poate analiza modul în care aceeași problemă de gestiune a datelor poate fi rezolvată atât prin comenzi SQL clasice, cât și prin instrumente moderne de analiză vizuală a datelor (de exemplu, Google Data Studio, Power BI).

Activitățile pot fi centrate pe proiecte de durată (individuale sau de grup) care implică proiectarea și gestionarea bazelor de date, dezvoltarea unor aplicații digitale, astfel încât elevii să poată experimenta întregul ciclu de dezvoltare al unei aplicații informatice – de la analiză și proiectare, până la testare și prezentarea rezultatelor.

Pentru clasa a XII-a se recomandă ca elevii să valorifice competențele formate pe parcursul claselor anterioare în cadrul unor proiecte complexe, interdisciplinare, axate pe rezolvarea de probleme reale și pe construirea de produse informatice funcționale, integrând instrumente de lucru colaborativ online.

Se recomandă realizarea unui proiect integrator, care ar putea implica realizarea unei baze de date pentru un scenariu dat, în care să se integreze, pe parcurs, elemente din ce în ce mai complexe, pe baza studiului conținuturilor specifice, asigurând documentarea, identificarea necesarului afacerii, realizarea modelului conceptual (cu elemente cât mai variate), maparea acestuia, implementarea unor module ale acestuia, precum și testarea și realizarea documentației aferente. O astfel de integrare ar sublinia coerența practică a conținuturilor predate și ar antrena competențele transversale.

Se recomandă o abordare activă, practică și exploratorie, în care profesorul are rolul de mentor, iar elevul devine creator de soluții, astfel încât procesul de învățare să cultive autonomia, gândirea critică și etica în utilizarea tehnologiei. Profesorul ghidează procesul de învățare prin întrebări, studii de caz și prin sprijin acordat în momentele-cheie, facilitând trecerea de la învățarea asistată la învățarea autonomă și reflexivă.

Se recomandă integrarea noțiunilor introductive de inteligență artificială în proiecte interdisciplinare. Elevii pot explora modul în care o bază de date alimentată cu exemple reale poate susține un model de învățare automată, pot reflecta critic asupra modului în care inteligența artificială influențează deciziile, discutând despre biasul datelor, confidențialitate și responsabilitate în contextul discuțiilor despre algoritmi utilizați în învățarea automată.

Se recomandă ca, la finalul clasei a XII-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 crește motivația elevilor și 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;
- UiPath Foundation (<https://aigeneration.uipathfoundation.com/ro/>), platformă ce conține planuri de lecție, prezentări, fișe, teste privind introducerea în inteligența artificială;
- pyML.ro (<https://www.pyml.ro/>), platformă cu lecții pentru introducere în învățare automată.

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, attribute, 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,

Nume și prenume	Grad didactic/Titlu științific	Instituție de apartenență, localitate, județ
		Facultatea de Automatică și Calculatoare
NIȚU Alina	profesor, gradul didactic I	Liceul Teoretic „Ovidius”, Constanța, județul Constanța
NURLA Aidan	profesor, gradul didactic I	Colegiul Național Militar „Alexandru Ioan Cuza”, Constanța, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forțelor 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 Național Militar „Dimitrie Cantemir”, Breaza, județul Prahova
TEGLAȘ Maria	profesor, gradul didactic II	Colegiul Național 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

Programa școlară pentru disciplina Informatică, clasa a XII-a, curriculum de specialitate (CS);
 filiera teoretică, profilul real, specializarea matematică-informatică; filiera vocațională, profilul militar, specializarea matematică-informatică militară

BORIGA Radu Eugen	conferențiar universitar, doctor	Universitatea București, Facultatea de Matematică și Informatică
-------------------	----------------------------------	--

Anexa nr. 5

Programa școlară
pentru disciplina

Informatică

Clasa a X-a

Curriculum de specialitate (CS)

*Filiera vocațională, profilul militar,
specializarea matematică-informatică militară*

Învățământ liceal

- 2026 –

Anexa nr. 5**Programa școlară pentru disciplina Informatică, clasa a X-a, curriculum de specialitate (CS)**

- *Filiera vocațională, profilul militar, specializarea matematică-informatică militară*

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 vocațională, profilul militar, specializarea matematică-informatică militară, alocarea orară este:

- clasa a IX-a – 3 ore/săptămână, dintre care o oră pentru studiu teoretic și două ore pentru activități practice;
- clasa a X-a – 3 ore/săptămână, dintre care o oră pentru studiu teoretic și două ore pentru activități practice;
- clasa a XI-a – 3 ore/săptămână, dintre care o oră pentru studiu teoretic și două ore pentru activități practice;
- clasa a XII-a – 3 ore/săptămână, dintre care o oră pentru studiu teoretic și două ore pentru 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 atestatorilor 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 limbajelor de programare Python și 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 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. 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 - neliniare, liniare, asociative, mixte, pentru a structura și accesa date în vederea prelucrării acestora	
- definirea conceptului de mulțime în contextul determinării drepturilor distincte de acces care sunt atribuite unui grup de utilizatori cu roluri diferite, unde fiecare drept apare o singură dată, chiar dacă e atribuit mai multor persoane, iar ordinea înregistrării acestora nu contează - enumerarea operațiilor de bază asupra mulțimilor (reuniune, intersecție, diferență, egalitate, incluziune, testare apartenență), în contextul în care se cunoaște lista obiectelor transportate cu două rucsacuri și se cere să fie identificate obiectele comune, cele conținute de un singur rucsac etc. - precizarea caracteristicilor unei mulțimi în contextul gestionării utilizatorilor înregistrați pe platforme diferite pentru a evita dublurile - 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 - 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 - definirea conceptului de structură asociativă (dicționar) cu evidențierea faptului că se rețin perechi cheie-valoare și se permite accesul rapid la date prin intermediul cheilor, de exemplu, într-o aplicație de traducere, în care fiecare cuvânt este o „cheie”, iar traducerea sa este „valoarea” sau în contextul asocierii dicționarului cu agenda telefonică în care fiecărui număr de telefon (cheie) îi corespunde un nume (valoare) - enumerarea cheilor și a valorilor asociate dintr-un dicționar pentru un formular online, în care fiecare câmp completat (de exemplu, „nume”, „vârstă”, „email”) este asociat cu valorile corespunzătoare - 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; - identificarea caracteristicilor unui model mixt, de tipul dicționar de liste și a modului de organizare a datelor în contextul evidenței școlare a claselor, în care fiecare clasă are o listă de elevi - enumerarea metodelor de acces și parcurgere a datelor într-o listă de dicționare în contextul gestiunii unor comenzi, fiecare comandă fiind reprezentată printr-un dicționar (prin asocierea produs-cantitate) pentru a genera facturile corespunzătoare - enumerarea caracteristicilor tuplurilor în contextul utilizării lor pentru stocarea coordonatelor geografice (latitudine, longitudine) într-un sistem de navigație care necesită date ce nu trebuie să fie modificate accidental - identificarea caracteristicii tuplurilor de reprezentare a unor perechi fixe de valori în contextul memorării culorilor ca triplete RGB, într-o aplicație de grafică - enumerarea caracteristicilor modelelor conceptuale liniare tuplu și listă, în contextul reprezentării unui punct fix de pe o hartă (de exemplu, un oraș, ale cărui coordonate nu pot fi modificate) și a unui punct mobil de pe hartă (care se actualizează, de exemplu, poziția unui robot care se mișcă)	
CS 1.2. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea listelor, a șirurilor de caractere	
- recunoașterea caracteristicilor metodei căutării binare cu specificarea condiției necesare, în contextul localizării rapide a unui contact într-o agendă telefonică ordonată alfabetic - enumerarea etapelor metodei căutării binare în contextul localizării unui titlu într-o listă ordonată de cărți dintr-o bibliotecă digitală - 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 - 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 - enumerarea unor metode simple de criptare (de exemplu, cifrul lui Cezar, substituția monoalfabetică) în contextul utilizării unor tehnici de codificare pentru a ascunde indicii într-un joc de tip „vânătoare de comori”	

Clasa a X-a	
- identificarea caracteristicilor specifice fiecărei metode de criptare (de exemplu, cifrul lui Cezar - deplasarea literelor, substituție monoalfabetică - înlocuire literă cu literă) în contextul analizării unui mesaj codificat, observând dacă literele sunt deplasate, înlocuite sau reordonate	
CS 1.3. Identifică specificul, caracteristicile și etapele unor strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor	
- definirea caracteristicilor metodei Divide et impera în contextul unei discuții ghidate în care sunt reliefate avantajele metodei prin analogie cu fragmentarea unei activități complexe în subactivități realizate de echipe diferite, urmate de reunirea rezultatelor - enumerarea pașilor specifici metodei Divide et impera (divizarea problemei, rezolvarea subproblemelor, combinarea soluțiilor) în contextul aplicării acestei strategii la algoritmul de determinare a valorii maxime dintr-un șir de numere memorate - 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 - 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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice	
- recunoașterea sintaxei corecte de inițializare a unei mulțimi (set) în contextul în care sunt adunate obiecte diferite (de exemplu, tipuri de fructe) într-o colecție în care elementele sunt unice, cu evidențierea modului în care utilizarea acoladelor și a elementelor separate prin virgule permite crearea unei mulțimi fără dubluri și fără ordine prestabilită - identificarea operatorului pentru verificarea apartenenței unui element la o mulțime în Python în contextul scrierii unei secvențe de instrucțiuni prin care se verifică dacă o persoană este deja înscrisă pe lista participanților la un eveniment, fără a fi necesară parcurgerea întregii liste - 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.) - 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 - 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 - 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 - recunoașterea notărilor specifice cu acolade și a separatorului între cheie și valoare în cazul dicționarelor, prin analogie cu organizarea informațiilor într-o listă de contacte: numele persoanei urmat de simbolul „:”, apoi de numărul de telefon, elementele fiind grupate între acolade - enumerarea operațiilor de bază asupra dicționarelor (creare, adăugare, ștergere, actualizare) prin analogie cu actualizarea listei de contacte din agenda telefonică: crearea unei liste de contacte, adăugarea unui contact nou, ștergerea unui contact existent, modificarea numărului de telefon al unei persoane - 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 - identificarea modului de inițializare a unui tuplu în Python, pentru un punct fix pe o hartă, dat prin coordonatele (x,y) - recunoașterea sintaxei de accesare a componentelor unui tuplu în Python, în contextul accesării abscisei sau ordonatei unui punct fix de pe o hartă, reprezentat printr-un tuplu	
CS 1.5. Identifică elementele de sintaxă și componentele din definiția și apelul subprogramelor recursive, pentru a le utiliza în implementarea algoritmilor în limbaj de programare	

Clasa a X-a
- enumerarea caracteristicilor principale ale unui subprogram recursiv, în contextul elaborării unui astfel de subprogram pentru calculul lui $n!$ definit prin relația recurentă cunoscută - recunoașterea termenilor „recursivitate directă” și „recursivitate indirectă” prin asocierea lor cu definițiile corespunzătoare - recunoașterea sintaxei corecte pentru definirea unui subprogram recursiv în Python, în contextul calculării sumei cifrelor unui număr - enumerarea condițiilor pe care trebuie să le îndeplinească un subprogram recursiv pentru a funcționa corect, menționând necesitatea unui caz de oprire, a unei transformări care reduce progresiv problema și a unui apel recursiv care aplică aceeași procedură până la atingerea condiției de oprire, în contextul calculării factorialului unui număr

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 - neliniare, liniare, asociative, mixte, pentru a le gestiona eficient
- explicarea caracteristicilor mulțimilor și listelor, în contextul organizării și accesării elementelor acestora în situații cotidiene, de exemplu, o listă de persoane care așteaptă la rând și determinarea clienților distincți ai unei companii de telefonie, pe baza unei mulțimi de abonamente (în condițiile în care un client poate avea mai multe astfel de abonamente) - explicarea modului în care mulțimile sunt utilizate pentru clasificarea pe categorii a produselor într-un magazin online, evidențiind cum apartenența la o categorie permite organizarea și filtrarea eficientă a articolelor - explicarea modului în care este prelucrat un șir de caractere, în contextul funcționării unui sistem de verificare a complexității parolilor, care analizează fiecare simbol pentru a identifica litere mari, litere mici, cifre și caractere speciale - 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 - explicarea rolului cheilor unice și a valorilor asociate în dicționare, în contextul serviciului de evidența populației, unde nu pot exista două CNP-uri identice (chei unice), iar modificarea numelui unei persoane actualizează valoarea corespunzătoare cheii respective - explicarea modului în care un dicționar poate fi folosit pentru a organiza toate informațiile unei comenzi online, fiecare cheie din dicționar reprezentând un tip de informație (de exemplu, „client”, „adresă”, „produse”), iar valoarea asociată acelei chei conținând detaliile respective, astfel încât toate datele comenzii să fie grupate logic și să fie ușor de găsit - 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; - explicarea modului în care un dicționar este folosit pentru a structura datele într-un sistem de gestiune a proiectelor, fiecare cheie reprezentând numele unei echipe, iar valoarea asociată fiind lista sarcinilor acelei echipe, ceea ce permite vizualizarea clară și separată a responsabilităților fiecărei echipe - explicarea modului în care structura unui orar poate fi reprezentată printr-un dicționar, în care fiecare cheie corespunde unei zile, iar valoarea asociată este lista orelor din acea zi, ceea ce permite accesul rapid la programul complet al unei anumite zile - explicarea caracteristicilor unui tuplu în contextul unei aplicații financiare, evidențiind motivul pentru care tuplurile sunt folosite pentru informații fixe (precum perechi de tipul „nume client – CNP”), astfel încât datele care nu trebuie schimbate să rămână protejate - explicarea modului în care tuplurile sunt adecvate pentru stocarea înregistrărilor fixe într-o aplicație de calendar, unde fiecare eveniment este reprezentat printr-un tuplu (data, ora, titlu_eveniment), ceea ce oferă o structură constantă și ușor de prelucrat pentru toate intrările
CS 2.2. Explică etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea listelor, a șirurilor de caractere
- 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 - 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, identificând eventualele suprapuneri) - 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

Clasa a X-a
(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 - explicarea modului de funcționare al diferitelor metode de criptare (de exemplu, cifrul lui Cezar, substituție simplă, cifrul Vigenère) în contextul protejării unui mesaj trimis online, evidențiind cum transformările aplicate textului urmăresc ascunderea semnificației lui - explicarea modului în care diferitele metode de criptare produc modificări specifice asupra unui șir de caractere, în contextul mesajelor codificate din jocuri de tip „vânătoare de comori”, evidențiind faptul că literele pot fi deplasate, înlocuite sau reordonate în funcție de metoda utilizată - descrierea modului în care metodele simple de criptare se aplică pe exemple concrete, de exemplu, folosirea cifrului lui Cezar pentru a transforma cuvântul „ANA” în „DQC” prin deplasarea fiecărei litere cu 3 poziții în alfabet, utilizarea unui cifru prin substituție pentru a codifica cuvântul „CARTE” folosind un tabel în care $C \rightarrow M$, $A \rightarrow Q$, $R \rightarrow T$, $T \rightarrow A$, $E \rightarrow Z$, respectiv aplicarea cifrului Vigenère pentru a cripta un mesaj cu un cuvânt cheie, obținând un șir în care fiecare literă este modificată adunând numărul de ordine din alfabet al literei din mesaj cu numărul de ordine al caracterului din cheia de criptare modulo 26
CS 2.3. Explică etapele unor strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor - explicarea etapelor metodei Divide et impera (descompunere, rezolvare, combinare), folosind ca exemplu calcularea sumei unor numere dintr-o listă, prin împărțirea repetată a acesteia și a fiecărei liste obținute pe parcurs în jumătăți (până când se ajunge la liste formate dintr-un singur număr, pentru care se cunoaște rezultatul) și determinarea rezultatului pentru o astfel de listă, prin adunarea sumelor corespunzătoare celor două jumătăți, după obținerea acestora - explicarea modului în care se aplică metoda Divide et impera, descriind clar cele trei etape ale sale — descompunerea problemei în subprobleme mai mici, rezolvarea fiecărei subprobleme, apoi combinarea rezultatelor în exemple reale în care această abordare este folosită, precum procesarea imaginilor sau navigația GPS - 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 - 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 $3+3$) - 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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice - explicarea modalității de parcurgere a elementelor unei mulțimi în Python prin analogie cu procesul de verificare a fiecărui obiect dintr-o cutie care conține piese unice, fără a ține cont de ordinea lor, evidențiind modul în care o instrucțiune repetitivă permite accesarea fiecărui element al mulțimii - explicarea efectului aplicării metodelor (<code>add()</code>, <code>remove()</code>, <code>union()</code>, <code>intersection()</code>) și operatorilor (<code> </code>, <code>&</code>, <code>-</code>) asupra a două mulțimi în Python, în contextul ilustrării în limbaj de programare a unor exemple inspirate din viața cotidiană (de exemplu, compararea listelor de prieteni din două rețele sociale, respectiv a prietenilor existenți în oricare rețea, a prietenilor comuni, a prietenilor prezenți doar în una dintre rețele) - explicarea efectului aplicării metodelor uzuale ale clasei <code>str</code>, în contextul ilustrării în limbaj de programare a unor exemple inspirate din viața cotidiană, de exemplu, separarea cuvintelor dintr-o propoziție (<code>split()</code>), transformarea textului în litere mici sau mari pentru uniformizare (<code>lower()</code>, <code>upper()</code>), eliminarea spațiilor nedorite de la începutul sau sfârșitul unui mesaj (<code>strip()</code>), înlocuirea unui cuvânt greșit (<code>replace()</code>), numărarea aparițiilor unui caracter într-un text (<code>count()</code>), verificarea faptului că un text conține doar litere (<code>isalpha()</code>) sau doar cifre (<code>isdigit()</code>) - explicarea efectului aplicării unor metode specifice dicționarilor prin asocierea acestora cu gestionarea informațiilor din agenda telefonică, de exemplu, verificarea faptului că un nume se află în agendă (<code>get()</code>), obținerea listei care conține doar numele din agendă (<code>keys()</code>), obținerea listei care conține doar numerele de telefon din agendă (<code>values()</code>), obținerea listei complete nume-număr din agendă (<code>items()</code>) - explicarea rezultatelor obținute în urma iterării asupra unui dicționar, cu exemplificare pe agenda telefonică, din care, în urma iterării, se poate obține doar lista de nume, doar lista de numere de telefon sau lista de perechi nume-număr de telefon, în funcție de metoda de parcurgere utilizată - 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 <code>i</code>, pe poziția <code>j</code>, este 1, dacă țările <code>i</code> și <code>j</code> sunt vecine și este 0, altfel

Clasa a X-a	
- explicarea efectului aplicării funcțiilor uzuale pentru prelucrarea tuplurilor în contextul memorării numărului de ore pentru activități extrașcolare din fiecare zi a săptămânii (de exemplu, ore_pe_zi = (3, 2, 4, 1, 5, 2, 0)) și aflarea numărului total de zile cu activități extrașcolare (len()), a celei mai aglomerate zile (max()), a zilei cu cele mai puține ore (min()), a numărului total de ore (sum()), a orelor în ordine crescătoare (sorted()), verificarea faptului că există cel puțin o zi cu ore programate (any()) sau că toate zilele reținute au ore programate (all()) - explicarea operației de extragere a elementelor unui tuplu în variabile (despachetare), în contextul salvării numărului de ore pentru activități extrașcolare în variabile aferente fiecărei zile din săptămână (de exemplu, dacă ore_pe_zi = (3, 2, 4, 1, 5, 2, 0), putem salva numărul de ore din fiecare zi astfel: luni, marti, miercuri, joi, vineri, sambata, duminica=ore_pe_zi) - explicarea efectului utilizării operatorilor pentru combinarea (+) și extinderea tuplurilor (*), în contextul combinării echipamentelor sportive pentru două echipe sau a extinderii unui tuplu care reprezintă meniul comandat de mai multe persoane într-o cantină	
CS 2.5. Explică mecanismul de executare a subprogramelor recursive, pentru a le utiliza în implementarea algoritmilor în limbaj de programare	
- explicarea modului de executare a unui subprogram recursiv, descriind cum fiecare apel este plasat pe stivă și rămâne în așteptare până la rezolvarea apelului următor, folosind ca exemplu o serie de cutii puse una în alta: întâi se deschide cutia cea mai mare, apoi cutia aflată în interiorul ei și tot așa până la cutia cea mai mică; abia după ce se ajunge la ultima cutie și se preia obiectul plasat în aceasta, cutiile sunt închise în ordinea inversă deschiderii, exact cum apelurile recursive sunt rezolvate în ordine inversă plasării lor pe stivă - explicarea modului în care un algoritm iterativ poate fi transformat într-o variantă recursivă echivalentă, folosind ca exemplu calcularea sumei primelor n numere naturale: varianta iterativă adună pe rând numerele de la 1 la n într-un ciclu, iar varianta recursivă calculează suma exprimând-o ca valoarea curentă plus suma numerelor rămase - explicarea modului în care apelul recursiv al unui subprogram conduce la reluarea instrucțiunilor acestuia, într-un alt context, mai restrâns, pentru rezolvarea unor probleme matematice precum calcularea factorialului, determinarea celui de-al n-lea termen al unui șir definit recursiv sau calculul celui mai mare divizor comun prin scăderi repetate, evidențiind necesitatea unui caz de oprire și a faptului că funcția se apelează repetat	

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 - neliniare, liniare, asociative, mixte, pentru a rezolva probleme de gestionare a datelor	
- aplicarea operațiilor de reuniune și intersecție asupra a două mulțimi de clienți din două magazine online, pentru a stabili atât lista tuturor clienților care pot primi oferte promoționale, cât și clienții comuni celor două magazine - utilizarea mulțimilor pentru generarea elementelor unice (de exemplu, coduri de acces fără repetiție), verificarea dublurilor într-o listă de elemente (de exemplu, produse identice într-un inventar), verificarea faptului că o colecție conține toate valorile dintr-o secvență de elemente (de exemplu, verificarea faptului că toate zilele unei săptămâni au fost completate într-un raport) - utilizarea comparării lexicografice pentru ordonarea titlurilor de cărți într-o bibliotecă digitală, astfel încât acestea să fie afișate alfabetic într-o listă - 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) - 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 - utilizarea organizării datelor sub formă de dicționar pentru realizarea unui glosar online care asociază niște termeni din tehnologia informației (chei) cu explicațiile lor (valori), facilitând învățarea rapidă a vocabularului de specialitate - utilizarea dicționarelor pentru organizarea perechilor cheie-valoare și obținerea unor rezultate, în contextul parcurgerii listei de produse dintr-un magazin unde fiecare produs (cheia) are asociat un preț (valoarea) și obținerea de rezultate precum valoarea totală a stocului, prețul minim/maxim, media prețurilor, produsele cu prețul mai mare/mai mic decât o valoare dată etc. - accesarea unui element dintr-un dicționar utilizat într-o aplicație de comerț online, în care codul unic al produsului este folosit drept cheie, iar valorile asociate sunt informațiile despre produs (denumire și preț), pentru a afișa rapid detaliile corespunzătoare codului introdus de utilizator - 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;	

Clasa a X-a
- utilizarea tuplurilor pentru reprezentarea coordonatelor geografice într-o aplicație de navigație GPS, prin memorarea fiecărui punct de interes într-un tuplu format din două valori fixe (latitudine, longitudine), astfel încât aceste perechi să poată fi folosite direct în calcule precum determinarea distanței dintre două locații - aplicarea unui algoritm de sortare asupra unei liste de produse reprezentate prin tupluri (denumire, preț), folosind al doilea element al fiecărui tuplu pentru a ordona produsele crescător după preț - utilizarea tuplurilor pentru calcularea mediei la o disciplină școlară, prin memorarea notelor unui elev în tupluri de forma (disciplină, notă), selectarea tuplurilor corespunzătoare acelei discipline, adunarea notelor și împărțirea sumei la numărul de note adunate pentru a obține media
CS 3.2. Aplică algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea listelor, a șirurilor de caractere
- 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 - 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 - 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 - 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 - aplicarea algoritmilor de criptare corespunzători metodelor studiate pentru transformarea unui mesaj obișnuit într-un mesaj criptat, folosind reguli precise de modificare a caracterelor, în scopul protejării informațiilor transmise - exersarea unui sistem de criptare a mesajelor folosind cifrul lui Cezar, prin introducerea unui text format din litere mari și a unei chei de criptare care indică numărul de poziții cu care fiecare literă este deplasată, obținând mesajul criptat - utilizarea metodei substituției simple pentru transmiterea de mesaje confidențiale între elevii unei clase, prin înlocuirea fiecărei litere din mesaj cu o altă literă stabilită conform unui tabel de substituție cunoscut, astfel încât mesajul rezultat să poată fi citit doar de destinatarii care cunosc regula de corespondență
CS 3.3. Aplică strategii de tip Divide et impera, 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 - 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) - aplicarea metodei de sortare prin interclasare pentru obținerea unei liste ordonate pornind de la o listă neordonată formată din 7 numere date - 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 - 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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice
- utilizarea unor operatori (, &, -) și metode (add(), remove(), union(), intersection()) ale clasei set în contexte practice, de exemplu, pentru gestionarea participării elevilor la activități extrașcolare, cu evidențierea modului în care aceste operații pot simplifica prelucrarea și compararea datelor - utilizarea unor metode adecvate din clasa set în implementarea unor programe pentru verificarea faptului că o listă este formată din elemente distincte (de exemplu, coduri unice de participare), eliminarea elementelor duplicate dintr-o listă (de exemplu, coduri introduse de mai multe ori), determinarea elementelor comune între două colecții (de exemplu, elevii înscriși la două cercuri diferite) - 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())

Clasa a X-a	
- 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()) - aplicarea metodelor uzuale ale clasei dict în contexte practice, de exemplu, pentru gestionarea produselor dintr-un magazin: actualizarea stocului de produse (update()), ștergerea unui articol din magazin (pop()), ștergerea ultimului produs adăugat (popitem()), lichidare de stoc (clear()), crearea unei copii de siguranță a stocului înainte de modificări majore (copy()), crearea unui nou stoc pornind de la un stoc existent (fromkeys()) în care vor fi toate produsele inițiale (cheile) dar prețurile aferente (valorile) au valoarea None, verificarea existenței unui produs în magazin cu scopul obținerii prețului acestuia și evitarea erorilor în cazul în care produsul lipsește (get()), adăugarea automată a unui produs în magazin dacă nu există deja (setdefault()), obținerea listelor cu denumirile produselor pentru inventar (keys()), prețurile din magazin (values()) sau perechile produs-preț (items()) pentru afișare pe etichete - utilizarea de dicționare imbricate, cu exemplificare pe gestionarea stocului unui magazin (reprezentat sub formă de dicționar) în care fiecare produs (cheia) are asociat un alt dicționar cu detalii ca preț, cantitate etc. - 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 - aplicarea mecanismului de despachetare a unui tuplu (tuple unpacking) pentru extragerea valorilor (de exemplu, integrarea într-o aplicație de gestionare a elevilor, în care funcțiile returnează mai multe informații) - aplicarea metodelor și funcțiilor uzuale pentru tupluri în rezolvarea unor probleme practice, de exemplu, pentru gestionarea unui magazin, în care încasările săptămânale sunt memorate în tupluri, și pe baza acestora se determină numărul total de zile cu încasări, ziua cu cele mai multe/puține încasări, suma totală încasată pe săptămână, încasările sortate crescător/descrescător, verificarea existenței a cel puțin unei zile cu încasări, verificarea faptului că au fost încasări în fiecare zi înregistrată	
CS 3.5. Utilizează elementele de sintaxă și componentele din definiția și apelul subprogramelor recursive, în implementarea algoritmilor în limbaj de programare	
- aplicarea elementelor de sintaxă pentru implementarea unui subprogram recursiv, pe baza unei relații recurente date, care determină factorialul unui număr, de exemplu, pentru stabilirea numărului de moduri diferite în care pot fi așezate 4 persoane pe 4 scaune într-o sală - aplicarea recursivității pentru determinarea unui termen din șirul lui Fibonacci, prin urmărirea apelurilor unui subprogram recursiv în Python - aplicarea recursivității pentru calcularea sumei elementelor unei liste, pe baza unui subprogram recursiv dat în limbajul Python, care adună elementul curent al listei cu rezultatul obținut prin apelarea sa asupra restului listei - implementarea unui subprogram recursiv pornind de la o relație de recurență dată, de exemplu, pentru calculul combinărilor de câte k elemente dintre cele n date, pe baza formulei $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ cu respectarea condițiilor matematice	

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 - neliniare, liniare, asociative, mixte, pentru a alege structura adecvată în rezolvarea unor probleme concrete	
- analizarea avantajelor și limitărilor utilizării mulțimilor în comparație cu listele, prin raportare la situații practice, precum verificarea rapidă a înscrierii unui elev în catalogul clasei, cu evidențierea faptului că verificarea apartenenței unui element este mai eficientă din punctul de vedere al numărului de operații efectuate, utilizând mulțimi comparativ cu liste, dar acestea nu păstrează ordinea și nu acceptă elemente duplicat	

Clasa a X-a

- 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
- 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)
- analizarea avantajelor utilizării unui dicționar într-o aplicație financiară, în care codul fiscal al fiecărui client este utilizat drept cheie pentru accesarea directă a datelor asociate, astfel încât informațiile despre un client sunt obținute imediat, fără a fi nevoie de verificarea succesivă a tuturor celorlalți clienți din evidență
- analizarea comportamentului datelor organizate în dicționare la actualizarea valorilor pentru chei existente versus chei noi, în contextul modificării prețului unui produs existent (cheie existentă) sau la adăugarea unui produs nou (cheie nouă) într-un magazin
- analizarea modului în care este utilizat un dicționar într-un sistem de evidență al unei biblioteci, în care fiecare carte este identificată printr-o cheie (de exemplu, codul cărții), iar valoarea asociată indică plasarea acesteia, pentru a verifica dacă fiecărei chei îi corespunde o plasare, evitând astfel pierderea informației despre plasarea cărților în bibliotecă
- 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
- analizarea adecvării utilizării tuplurilor, pentru a organiza datele (disciplinele) în cadrul unui program săptămânal fixat
- examinarea modului în care folosirea tuplurilor într-o aplicație medicală contribuie la păstrarea neschimbată a datelor esențiale ale unui pacient (nume, CNP, data nașterii), prin utilizarea unei structuri cu valori fixe care limitează alterarea accidentală a informațiilor
- analizarea modului în care perechile (produs, preț) sunt memorate sub formă de tupluri într-un sistem de vânzări, pentru a verifica dacă prețul asociat fiecărui produs este corect utilizat în prelucrări precum afișarea costului unui articol, calculul valorii totale a unei comenzi sau compararea prețurilor între produse

CS 4.2. Analizează comportamentul, aplicabilitatea, avantajele și dezavantajele utilizării unor algoritmi specializați pe clase de probleme pentru prelucrarea listelor, a șirurilor de caractere

- 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
- 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
- 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
- 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
- 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
- compararea metodelor de criptare studiate (cifru lui Cezar, substituția simplă, cifrul Vigenère) din perspectiva modului de transformare a mesajului, a tipului de cheie utilizate și a nivelului de dificultate în descifrarea unui mesaj real, precum transmiterea unui mesaj confidențial între două persoane
- analizarea mesajelor criptate obținute în situații concrete cu metodele studiate, precum trimiterea unui cod de acces sau a unui mesaj privat, pentru a identifica erori de criptare, cum ar fi litere transformate incorect, pierderea unor caractere sau aplicarea greșită a regulilor metodei aale
- examinarea asemănărilor și deosebirilor dintre metodele de criptare utilizate în contexte reale, precum comunicarea mesajelor secrete sau protejarea informațiilor scrise, prin raportare la modul de transformare a literelor, dependența de cheie și ușurința cu care mesajul poate fi descifrat

CS 4.3. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor

- analizarea aplicării metodei Divide et impera printr-un exemplu concret, în care o listă de scoruri ale jucătorilor este împărțită repetat în jumătate, până când se ajunge la liste formate dintr-un singur scor, fiecare sublistă este prelucrată separat prin aceeași regulă (de exemplu, determinarea scorului maxim/minim din sublistă), iar rezultatele obținute sunt apoi comparate pentru a stabili scorul maxim din lista inițială, evidențiind explicit etapele de împărțire, rezolvare și combinare

Clasa a X-a	
- <i>analizarea procesului de împărțire succesivă a unei probleme prin metoda Divide et impera, printr-un exemplu precum calculul sumei elementelor unei liste</i> - <i>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</i> - <i>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</i> - <i>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</i>	
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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice	
- <i>analizarea mesajelor de eroare apărute în Python la includerea într-o mulțime a unor elemente mutabile (de exemplu, liste sau dicționare), prin asocierea cu situația în care într-o colecție de obiecte unice se încearcă adăugarea unui obiect care își poate schimba forma (de exemplu, o trusă de geometrie ce poate conține doar reprezentări ale unor corpuri geometrice care sunt rigide, deci nu se pot modifica, comparativ cu cazul în care acestea ar fi realizate din plastilină, astfel încât să poată fi remodelate)</i> - <i>analizarea comportamentului unor programe care realizează operații de bază cu mulțimi (de exemplu, verificări rapide ale apartenenței, identificarea dublurilor într-o listă de înscrieri), utilizând clasa set, respectiv list, pentru alegerea unei variante eficiente din punctul de vedere al timpului de executare</i> - <i>analizarea avantajelor utilizării unei liste de șiruri de caractere, comparativ cu un șir compact, pentru afișarea în ordine alfabetică a unor cuvinte</i> - <i>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</i> - <i>analizarea avantajelor utilizării dicționarilor în contextul comparării stocurilor a două magazine pentru a identifica produse disponibile în ambele magazine (chei comune), produse care există doar într-un magazin și nu în celălalt (diferențe)</i> - <i>analizarea modului în care pot fi combinate dicționarele, în contextul unificării stocurilor din mai multe magazine, pentru adăugarea sau actualizarea datelor produselor din magazinul principal (update()), unificarea stocurilor pentru generarea unui catalog de produse (operatorii “ ”, “**”), unificarea manuală și verificarea cheilor duplicate, filtrarea produselor cu preț mai mare decât o valoare dată în momentul unificării</i> - <i>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</i> - <i>analizarea avantajelor utilizării unui tuplu care conține obiecte mutabile (liste, dicționare), în contextul reținerii și prelucrării datelor despre produsele dintr-un magazin într-un tuplu de liste, de exemplu, magazin=(['paine', 20], ['lapte', 15], ['mere', 30]), unde pe prima poziție a fiecărei liste este denumirea produsului, iar pe a doua poziție este cantitatea aferentă</i>	
CS 4.5. Analizează aplicabilitatea, avantajele și dezavantajele utilizării subprogramelor recursive, în implementarea algoritmilor în limbaj de programare	
- <i>analizarea consumului de timp și memorie al unui subprogram recursiv în Python care rezolvă o problemă matematică, precum determinarea sumei primelor n numere naturale, prin observarea numărului de apeluri recursive efectuate și a spațiului ocupat în stiva de executare pe măsură ce parametrul utilizat se modifică</i> - <i>compararea abordării recursive și a celei iterative în Python pentru rezolvarea unei probleme matematice precum calculul sumei primelor n numere naturale, analizând diferențele dintre cele două moduri de rezolvare în ceea ce privește numărul de pași efectuați, modul de repetare a operațiilor și utilizarea memoriei în funcție de valoarea lui n</i> - <i>compararea a doi algoritmi pentru determinarea unui termen al șirului lui Fibonacci, unul iterativ, iar celălalt recursiv, pe baza desfășurării pas cu pas a executării acestuia, urmărind ordinea apelurilor și revenirea din acestea, prin analogie cu verificarea succesivă a unor cutii puse una în alta, unde fiecare verificare nouă este memorată temporar până când se ajunge la ultima cutie și se revine treptat la prima, observând reluarea unor calcule</i>	

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 - neliniare, liniare, asociative, mixte și a modurilor de prelucrare a acestora pentru rezolvarea unor probleme informatice concrete	
- argumentarea adecvării utilizării mulțimilor care gestionează date neordonate și fără elemente duplicate (de exemplu, administrarea codurilor unice ale produselor dintr-un magazin, evidența participanților la un eveniment) cu justificarea alegerii acestora pentru avantajele de unicitate - evaluarea avantajelor utilizării mulțimilor în organizarea unui volum mare de informații despre clienți, în comparație cu alte forme de organizare precum listele, justificând alegerea mulțimilor prin capacitatea lor de a elimina automat aparițiile repetate și de a permite identificarea rapidă a elementelor unice, ceea ce reduce erorile și timpul de prelucrare a datelor - 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 - 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 - evaluarea corectitudinii utilizării unui dicționar digital într-o aplicație reală (de exemplu, un catalog electronic de produse), prin verificarea faptului că fiecare cheie este unică și că valoarea asociată conține informații complete și corecte, justificând că această structură previne confuziile și erorile de afișare în timpul utilizării - justificarea alegerii dicționarului pentru rezolvarea unor probleme din context cotidian (de exemplu, gestionarea unui catalog de elevi, a unei biblioteci, a unui concurs sportiv, a unei agenții de turism, organizarea informațiilor economice cum ar fi asocierea codurilor de factură cu datele corespunzătoare ale clienților), prin accesul rapid la date și prin claritatea relației dintre identificator și informația asociată - evaluarea avantajelor utilizării unei structuri de tip dicționar care asociază termeni cu definițiile lor (de exemplu, un glosar digital), prin aprecierea corectitudinii relației termen–definiție și justificarea modului în care această organizare sprijină regăsirea rapidă și fără ambiguități a informațiilor - evaluarea corectitudinii datelor într-un model conceptual mixt (de exemplu, listă de liste, listă de dicționare) 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) - argumentarea alegerii unui model mixt pentru un sistem de catalog electronic (de exemplu, listă de liste, listă de dicționare) pentru a organiza datele despre clase, elevi și note, asigurând o structură clară și flexibilă - evaluarea adecvării utilizării tuplurilor pentru memorarea unor date care nu se modifică, precum coordonatele geografice ale unui punct sau datele de identificare ale unui document, justificând alegerea prin stabilitatea valorilor și reducerea riscului de modificări accidentale - justificarea alegerii unui tuplu în locul unei liste pentru a organiza date constante (de exemplu, coordonatele unui punct fix pe hartă, programul fix al săptămânii, coduri de culori RGB etc.)	
CS 5.2. Evaluează corectitudinea și eficiența soluțiilor cu algoritmi specializați pe clase de probleme pentru prelucrarea listelor, a șirurilor de caractere	
- 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 - 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 - 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 - evaluarea adecvării metodelor de criptare și decriptare utilizate într-un context real, precum protejarea mesajelor transmise într-o aplicație de comunicare, prin justificarea alegerii metodei (cifrul lui Cezar, substituție simplă, cifrul Vigenère) în funcție de nivelul de securitate necesar, tipul cheii folosite și ușurința de aplicare - evaluarea eficienței utilizării metodelor de criptare într-un context real de transmitere a documentelor, de exemplu, trimiterea prin e-mail a unui fișier cu date personale, prin analizarea modului în care textul este criptat cu cifrul Vigenère pentru confidențialitate, în vederea justificării faptului că informația este protejată împotriva accesului neautorizat	
CS 5.3. Evaluează corectitudinea și eficiența algoritmilor care utilizează strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor	

Clasa a X-a	
- evaluarea eficienței metodei <i>Divide et impera</i> în rezolvarea unor probleme cotidiene (de exemplu, sortarea unui șir de melodii după durată sau căutarea rapidă a unui artist într-o listă ordonată alfabetic), prin justificarea alegerii metodei pe baza timpului de executare, a clarității pașilor și a comparării cu alte metode de rezolvare - evaluarea oportunității aplicării metodei <i>Divide et impera</i> în contexte matematice (de exemplu, căutarea unui element într-un interval numeric prin înjumătățiri succesive), prin justificarea deciziei pe baza modului de împărțire a problemei în subprobleme independente, a reducerii numărului de pași și a eficienței soluției finale - evaluarea corectitudinii unui algoritm bazat pe metoda <i>Divide et impera</i> utilizat într-o problemă matematică (de exemplu, calculul rapid al unei puteri sau al maximului dintr-un șir de valori), prin justificarea validității soluției obținute pentru toate subproblemele și a modului în care acestea sunt combinate - evaluarea modului în care optimul local în metoda <i>Greedy</i> 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 - evaluarea obținerii soluției optime pentru probleme rezolvate cu metoda <i>Greedy</i> (de exemplu, organizarea unui program de întâlniri fără suprapuneri), prin justificarea corectitudinii condițiilor alese - argumentarea deciziei de aplicare a metodei <i>Greedy</i> 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, $\text{card}(A)=n$ și $\text{card}(B) \geq n$), 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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice	
- evaluarea corectitudinii și eficienței diferitelor soluții de eliminare a elementelor duplicate dintr-o listă, prin compararea implementărilor iterative cu cele bazate pe conversia listei într-o mulțime (set), pentru a evidenția avantajele utilizării mulțimilor în simplificarea și accelerarea procesului de filtrare - evaluarea eficienței a două soluții pentru aceeași problemă, una care integrează operații specifice clasei set pentru îmbunătățirea clarității și performanței și alta bazată pe liste (de exemplu, gestionarea datelor de înscriere la un eveniment, filtrarea elementelor unice dintr-un fișier) - evaluarea eficienței și clarității redactării a două soluții pentru aceeași problemă, care verifică dacă un cuvânt este anagramă a unui alt cuvânt - argumentarea alegerii unei metode din clasa str pentru o sarcină practică, de exemplu, căutarea unor vocale într-un text dat - evaluarea eficienței unei soluții bazate pe dicționar față de una bazată pe listă de liste (listă de perechi), în contextul căutării rapide a prețului unui produs într-un catalog - evaluarea unui cod care folosește inadecvat metodele clasei dict (de exemplu, încercarea de a accesa o cheie inexistentă fără metoda <code>get()</code>, utilizarea metodei <code>pop()</code> cu argument greșit, încercarea de a modifica o cheie etc.) - 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 - evaluarea clarității unui cod care folosește tupluri pentru transmiterea mai multor valori dintr-o funcție (de exemplu, returnarea simultană a datelor despre un produs dintr-un magazin, cum ar fi denumire, preț, cantitate) - evaluarea corectitudinii unui cod în care sunt utilizate tupluri, având în vedere enunțul problemei corespunzătoare și corectarea eventualelor greșeli	
CS 5.5. Evaluează corectitudinea și eficiența programelor care utilizează subprograme recursive, pentru a rezolva probleme informatice	
- evaluarea eficienței unui subprogram recursiv în Python în raport cu o implementare iterativă echivalentă pentru o problemă matematică (de exemplu, calculul factorialului unui număr), prin justificarea diferențelor de claritate a soluției, număr de pași de executare și consum de memorie - evaluarea limitărilor în utilizarea unui subprogram recursiv în Python pentru rezolvarea unei probleme matematice (de exemplu, determinarea celui de al n-lea termen al șirului lui Fibonacci), evidențiind apeluri repetate pentru realizarea unor calcule identice - evaluarea deciziei de folosire a unui subprogram recursiv în Python pentru organizarea unui buget complex (de exemplu, calcularea soldului lunar pentru un buget structurat ierarhic, folosind liste de liste sau dicționare de dicționare), prin justificarea modului în care structura recursivă simplifică tratarea nivelurilor multiple de date față de o soluție iterativă	

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 - neliniare, liniare, asociative, mixte, pentru organizarea și prelucrarea eficientă a datelor, utilizând algoritmi de bază, în vederea rezolvării unor probleme	
- proiectarea unui algoritm de păstrare a evidențelor pentru o organizație educațională (de exemplu, o platformă de cursuri), pentru a identifica participanții înscriși, a evita înscrierile redundante, pe baza datelor utilizatorilor care sunt organizate în mulțimi - proiectarea unui algoritm pentru analizarea datelor unui magazin online, în care sunt definite explicit mulțimea tuturor clienților înregistrați și, pentru fiecare produs, mulțimea clienților care l-au achiziționat, iar relațiile dintre mulțimi sunt utilizate pentru recomandarea de perechi de produse care sunt achiziționate de obicei împreună - 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 - 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 - proiectarea unui algoritm de actualizare automată a unui dicționar de cursuri valutare, în care cheia este codul monedei (de exemplu, EUR, USD), iar valoarea este cursul de schimb asociat în lei, algoritmul preluând periodic date noi și înlocuind valorile existente cu cele actualizate - proiectarea unui sistem de înlocuire automată a termenilor dintr-un text, folosind un dicționar în care cheia este cuvântul sau expresia căutată, iar valoarea este cuvântul sau expresia care o înlocuiește, algoritmul parcurgând textul și substituind fiecare apariție a cheilor cu valorile corespunzătoare - proiectarea unui algoritm care utilizează un model mixt (listă de liste sau dicționar 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 - proiectarea unui algoritm de filtrare a comenzilor dintr-o listă în care pentru fiecare înregistrare se păstrează suma plătită și numărul bonului, pentru a afișa doar comenzile care depășesc o sumă prestabilită - proiectarea unui algoritm bazat pe tupluri pentru afișarea clasamentelor sportive, în care fiecare tuplu conține numele jucătorului, punctajul obținut și poziția finală, astfel încât datele să corespundă exact rezultatelor stabilite la finalul competiției și să nu poată fi modificate ulterior, păstrând coerența datelor pe toată durata competiției - proiectarea unui algoritm pentru înregistrarea itinerariului parcurs de un grup de turiști, utilizând tupluri, fiecare conținând coordonatele GPS și denumirea obiectivului vizitat	
CS 6.2. Proiectează soluții cu aplicarea personalizată a algoritmilor specializați pe clase de probleme pentru prelucrarea listelor, a șirurilor de caractere	
- 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 - 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 - 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 - proiectarea unor soluții informatice pentru protejarea mesajelor transmise într-un context real (de exemplu, comunicarea dintre membrii unei echipe sau stocarea unor note personale), prin proiectarea unor aplicații care implementează criptarea și decriptarea șirurilor de caractere folosind cifrul lui Cezar sau cifrul Vigenère, explicând modul în care cheia și regulile de transformare asigură confidențialitatea informației - proiectarea unor soluții pentru verificarea integrității datelor textuale (de exemplu, validarea unui fișier trimis online sau detectarea modificărilor unui mesaj), prin aplicarea algoritmului lui Fletcher, astfel încât soluția să fie mai eficientă și mai sigură în raport cu problema reală rezolvată	
CS 6.3. Proiectează algoritmi cu aplicarea personalizată a strategiilor de tip Divide et impera, Greedy pentru rezolvarea problemelor	
- proiectarea unor algoritmi care utilizează metoda Divide et impera, pentru contexte reale (de exemplu, căutarea unui document într-o arhivă ordonată), prin definirea adecvată a datelor și a subproblemelor, astfel încât rezultatul să fie obținut împărțind problema inițială în subprobleme mai mici de același tip - proiectarea unor algoritmi pentru crearea unor fractali simpli (de exemplu, covorul lui Sierpinski), utilizând metoda Divide et impera	

Clasa a X-a
- proiectarea unor soluții informatice care aplică personalizat metoda Divide et impera pentru problema turnurilor din Hanoi prin reprezentarea configurației tijelor folosind structuri de date adecvate și prin afișarea etapizată a mutărilor efectuate - 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 - 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 - neliniare, liniare, asociative, mixte, în vederea realizării unor soluții funcționale și performante
- implementarea unor aplicații personalizate care utilizează clase Python specifice pentru gestionarea mulțimilor la matematică (de exemplu, un software educațional cu titlul „Mulțimi”) sau generarea unor numere distincte pentru jocuri de tip Bingo - implementarea unor aplicații personalizate pentru prelucrarea cifrelor unui număr utilizând adecvat clasa set din Python (de exemplu, determinarea cifrelor distincte ale unui număr, formarea unui număr maxim cu cifrele distincte ale unui număr dat, determinarea cifrelor comune ale unor numere date, determinarea cifrelor care nu apar în scrierea unui număr etc.) - 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’) - 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) - implementarea unor programe personalizate folosind funcționalitățile clasei dict, pentru gestiunea stocurilor dintr-un magazin - implementarea unor programe personalizate folosind funcționalitățile clasei dict, pentru traducerea automată a unor cuvinte în mai multe limbi - 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 - integrarea tuplurilor în structuri mixte, cum ar fi liste de tupluri (de exemplu, pentru gestionarea stocului de produse într-un magazin, reprezentat sub formă de listă, în care fiecare produs este reținut printr-un tuplu care conține date fixe, cum ar fi codul produsului, denumirea, prețul), dicționare în care valorile sunt tupluri (de exemplu, catalog în care cheile sunt reprezentate de numele elevilor, iar valorile sunt notele fiecărui elev, reținute în tupluri) - implementarea unor aplicații care folosesc tupluri, de exemplu, pentru gestionarea punctajelor la un concurs, unde fiecare tuplu conține (nume, scor), pentru gestionarea coordonatelor geografice pentru un explorator sau pentru gestionarea codurilor de culori RGB pentru imagini
CS 6.5. Implementează aplicații în limbaj de programare care integrează subprograme recursive personalizate, pentru a modulariza și organiza eficient codul în cadrul soluțiilor informatice
- crearea unei aplicații pentru prelucrarea cifrelor unui număr care afișează mai multe opțiuni, iar pentru fiecare opțiune se apelează câte un subprogram recursiv (de exemplu, pentru afișarea cifrelor în ordine de la stânga la dreapta, de la dreapta la stânga, determinarea cifrei maxime, a numărului de cifre, a sumei cifrelor pare, verificarea existenței unei cifre pare, verificarea proprietății de platou format cu cifre egale), fiecare subprogram fiind elaborat de câte un membru al unei echipe de elevi - crearea unei aplicații cu subprograme recursive pentru prelucrarea unei liste ordonate de numere (de exemplu, mediile unor elevi), prin care să se afișeze numerele atât în ordine crescătoare, cât și în ordine descrescătoare - crearea unui subprogram recursiv pentru a verifica dacă un cuvânt este o anagramă a unui alt cuvânt, în contextul unei aplicații de jocuri enigmatice - crearea unei aplicații cu subprograme recursive care să afișeze pe ecran diverse forme geometrice simple cu ajutorul spațiilor și al unor simboluri (de exemplu, un triunghi format din simboluri *), în contextul unui concurs de realizare a designului unei sigle

CONȚINUTURI ALE ÎNVĂȚĂRII

Clasa a X-a

Domenii de conținut	Conținuturi
1. Organizarea conceptuală a datelor	1.1. Modelul conceptual neliniar - mulțime - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.2. Modelul conceptual liniar – șir de caractere - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.3. Modelul conceptual asociativ – dicționar - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.4. Modelul conceptual liniar - tuplu - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.5. Modelul conceptual mixt (de exemplu, liste de liste, liste de șiruri de caractere) - 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 - caracteristici ale metodei căutării binare și repere pentru aplicarea acesteia; - caracteristici ale metodei interclasării și repere pentru aplicarea acesteia. 2.2. Metode simple de criptare a șirurilor de caractere și verificare a integrității datelor - caracteristici ale criptării și decriptării folosind cifrul lui Cezar și repere pentru aplicarea metodei; - caracteristici ale criptării și decriptării prin metoda substituției simple (substituție monoalfabetică); - caracteristici ale criptării și decriptării utilizând cifrul Vigenère și repere pentru aplicarea metodei; - caracteristici ale sumei de control a unui șir de caractere sau a unui fișier text și determinarea sa utilizând algoritmul lui Fletcher. 2.3. Metoda Divide et impera - caracteristici ale metodei Divide et impera; - repere pentru aplicarea metodei; - caracteristici ale metodei de sortare prin interclasare și repere pentru aplicarea acesteia; - caracteristici ale metodei de sortare rapidă și repere pentru aplicarea acesteia. 2.4. Metoda Greedy - caracteristici ale metodei Greedy; - repere pentru aplicarea metodei.
3. Memorarea datelor și organizarea codului în limbaj de programare	3.1. Clasa set din Python – clasă predefinită pentru mulțimi - caracteristici; - operații și operatori specifici: reuniune, intersecție, diferență, egalitate, incluziune, apartenență; - metode uzuale pentru adăugare, ștergere, inițializare; - repere pentru identificarea și utilizarea altor metode adecvate pentru prelucrarea mulțimilor, în funcție de nevoile proprii. 3.2. Clasa str din Python – clasă predefinită pentru șiruri de caractere - caracteristici; - operații și operatori specifici: apartenență, concatenare, multiplicare, acces la un caracter sau subșir/secvență de caractere, comparare lexicografică; - 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; - repere pentru identificarea și utilizarea altor metode și funcții adecvate pentru prelucrarea șirurilor de caractere, în funcție de nevoile proprii. 3.3. Clasa dict din Python – clasă predefinită pentru dicționare - caracteristici; - operații și operatori specifici: acces la date;

Domenii de conținut	Conținuturi
	- repere pentru identificarea și utilizarea altor metode sau operatori adecvați pentru prelucrarea dicționarelor, în funcție de nevoile proprii. 3.4. Clasa tuple din Python – clasă predefinită pentru tupluri - caracteristici; - operații și operatori specifici: acces la date, apartenență, concatenare, multiplicare; - metode uzuale pentru numărare, căutare; - repere pentru identificarea și utilizarea altor clase și metode adecvate pentru prelucrarea tuplurilor, în funcție de nevoile proprii. 3.5. Elemente de limbaj pentru operarea cu date organizate în modele mixte - caracteristici, acces la un element în Python. 3.6. Subprograme recursive - caracteristici, rol, mecanism de executare; - repere pentru scrierea subprogramelor recursive în Python.

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 măsoară? 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.

Un context de învățare poate viza simultan mai multe competențe, mai ales atunci când acestea sunt interconectate. De exemplu, pentru rezolvarea unei probleme cu ajutorul calculatorului se vizează, în cadrul aceleiași activități de învățare, atât competențe de

nivel cognitiv scăzut (identificarea principalelor caracteristici ale modelelor conceptuale și operaționale implicate, explicarea principiilor care stau la baza acestora, în contextul particular al problemei, utilizarea lor în rezolvare) dar și competențe de nivel cognitiv ridicat (analizarea caracteristicilor modelelor conceptuale și operaționale specifice problemei, evaluarea corectitudinii și eficienței soluțiilor propuse și elaborarea algoritmului personalizat).

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 algoritmi 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 metode/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 matematică-informatică militară

1. *Strategii de rezolvare a problemelor. Metode de prelucrare a listelor ordonate*
2. *Organizarea conceptuală a datelor. Modelul conceptual neliniar - mulțime*
3. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa set din Python*
4. *Organizarea conceptuală a datelor. Modelul conceptual liniar – șir de caractere*
5. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa str din Python*
6. *Strategii de rezolvare a problemelor. Metode simple de criptare a șirurilor de caractere și verificare a integrității datelor*
7. *Organizarea conceptuală a datelor. Modelul conceptual asociativ – dicționar*
8. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa dict din Python*
9. *Organizarea conceptuală a datelor. Modelul conceptual liniar – tupluri*
10. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa tuple din Python*
11. *Organizarea conceptuală a datelor. Modelul conceptual mixt*
12. *Memorarea datelor și organizarea codului în limbaj de programare. Subprograme recursive*
13. *Strategii de rezolvare a problemelor. Metoda Divide et impera*
14. *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 conceptului de mulțime reprezintă o punte naturală între gândirea matematică și cea informatică. Se recomandă utilizarea strategiei descoperirii dirijate, pornind de la situații cotidiene, de exemplu, „mulțimea elevilor din clasă”, „mulțimea culorilor preferate”, pentru a stimula înțelegerea noțiunii de element și apartenență. Se pot folosi diagrame Venn pentru reprezentarea relațiilor de reuniune, intersecție și diferență, susținând astfel dezvoltarea gândirii vizuale și logice. Se recomandă integrarea metodei comparației pentru a reliefa caracterul neliniar al acestei structuri față de listele ordonate. Profesorul concretizează teoria mulțimilor prin clasa predefinită set. Se recomandă demonstrația dirijată și învățarea activă: elevii experimentează singuri comportamentul metodelor (add, remove, union, intersection, difference). Se valorifică interdisciplinaritatea cu matematica, punând accent pe semnificația logică a operațiilor.

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.

Algoritmii de criptare au un rol formativ interdisciplinar, îmbinând concepte de informatică, logică, matematică și securitate a datelor. Tema oferă o introducere intuitivă în criptografie și în ideea de protejare a informației, consolidând totodată lucrul cu șiruri de caractere și prelucrări pe baza codurilor ASCII/Unicode.

Se urmărește formarea gândirii algoritmice aplicate și înțelegerea principiului de transformare a datelor (criptare ↔ decriptare), nu memorarea formulelor. Abordarea metodologică trebuie să fie intuitivă, progresivă și comparativă, pornind de la concepte simple (cifru lui Cezar) către metode mai complexe (cifru Vigenère), accentuând caracteristicile conceptuale și logice ale algoritmilor, principiile comune ale criptării: existența unui text clar și a unui text cifrat, utilizarea unei chei sau a unei reguli de transformare și

posibilitatea de reversibilitate (criptare ↔ decriptare). Se recomandă utilizarea exemplelor concrete de mesaje și transformări simple pentru a formula regulile generale ale metodelor. Fiecare metodă se exprimă prin pași logici de prelucrare, accentuând legătura dintre idei și implementarea algoritmică și se valorifică legăturile cu alte domenii (matematică – operații modulo, statistică – frecvențe de apariție, TIC – fișiere text). Fiecare metodă poate fi analizată comparativ, în raport cu tipul transformării (substituție, deplasare, control), complexitatea algoritmului și gradul de securitate indus, evidențiind asemănările și deosebirile dintre acestea (cifrul lui Cezar, cifrul Vigenère, substituție).

Profesorul poate introduce modelul asociativ, reprezentat de dicționar, prin metoda modelării, explicând principiul perechilor cheie–valoare și apelând la exemple familiare (agenda de contacte, glosar de termeni). Elevii sunt ghidați prin conversație euristică și comparare între listă și dicționar, pentru a înțelege diferențele privind accesul la date. Se recomandă integrarea live coding-ului și a vizualizării în timp real a modificării structurilor, consolidând astfel legătura între teorie și practică.

Abordarea modelului conceptual liniar - tuplu poate porni de la exemple care să ilustreze ideea de colecție ordonată și imutabilă, cum ar fi coordonatele unui punct fix pe o hartă sau programul constant al unei zile. Pentru o mai bună înțelegere, se pot realiza paralele între tupluri și liste, prin analogii concrete: tuplul ca o informație permanentă (de exemplu, locația unei clădiri), iar lista ca o informație care se poate modifica (de exemplu, poziția unui robot). Compararea comportamentului tuplurilor și al listelor, din perspectiva mutabilității, performanței și siguranței datelor, evidențiind avantajele folosirii tuplurilor pentru informații fixe, poate contribui la o mai bună înțelegere a conceptelor. Se recomandă utilizarea exemplelor vizuale și contextuale (coordoanate, culori RGB, programul zilnic al unei clase etc.) pentru a evidenția modul de creare, accesare și parcurgere a elementelor unui tuplu, exerciții practice care implică prelucrarea datelor reale (de exemplu, reținerea și afișarea numărului de ore de activități zilnice, combinarea tuplurilor pentru generarea unui orar extins, extragerea valorilor multiple returnate de o funcție prin despachetare). Profesorul poate valorifica situații de învățare interdisciplinare (matematică, geografie, economie) pentru a consolida înțelegerea rolului tuplurilor în stocarea datelor constante.

Predarea modelelor mixte de organizare a datelor (liste de liste, liste de șiruri de caractere, liste de dicționare etc.) poate fi abordată gradual, pornind de la exemple simple și concrete (liste de liste, liste de dicționare), în care apare nevoia de ierarhizare și acces multiplu la date, către generalizări (dicționare de dicționare). 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.

Profesorul poate introduce conceptul de recursivitate folosind metoda analogiei („oglină într-o oglindă”) și reprezentarea grafică a apelurilor recursive. Este utilă învățarea prin simulare, pentru a vizualiza mecanismul stivei de apeluri. Metacogniția are un rol major: elevii trebuie să descrie verbal condițiile de oprire și fluxul execuției.

Pentru metoda Divide et impera, profesorul poate să introducă ideea de descompunere a problemei prin problematizare și euristică. Se recomandă construirea treptată a conceptului prin exemple familiare (găsirea elementului maxim, căutarea unui element, sortarea unei liste). Diagramele arborescente și vizualizarea recursivității facilitează înțelegerea structurii logice a algoritmului. O strategie eficientă este metacogniția – elevii sunt invitați să explice verbal modul de împărțire și combinare a subproblemelor. La predare, se evidențiază aplicarea acestei strategii și pentru metoda căutării binare, cu eliminarea uneia dintre subproblemele generate, având în vedere că nu influențează rezultatul. Se recomandă abordarea unor rezolvări cu aplicarea algoritmilor de bază (de exemplu, sume, minime), probleme clasice (de exemplu, problema turnurilor din Hanoi, problema covorului lui Sierpinski) ca model, apoi realizarea unor probleme asemănătoare care presupun adaptări, reinterpretări, extinderi.

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.

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://repl.it.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,

Nume și prenume	Grad didactic/Titlu științific	Instituție de apartenență, localitate, județ
		Facultatea de Automatică și Calculatoare
NIȚU Alina	profesor, gradul didactic I	Liceul Teoretic „Ovidius”, Constanța, județul Constanța
NURLA Aidan	profesor, gradul didactic I	Colegiul Național Militar „Alexandru Ioan Cuza”, Constanța, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forțelor 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 Național Militar „Dimitrie Cantemir”, Breaza, județul Prahova
TEGLAȘ Maria	profesor, gradul didactic II	Colegiul Național 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

Anexa nr. 6

Programa școlară
pentru disciplina

Informatică

Clasa a XI-a

Curriculum de specialitate (CS)

*Filiera vocațională, profilul militar,
specializarea matematică-informatică militară*

Învățământ liceal

- 2026 –

Anexa nr. 6**Programa școlară pentru disciplina Informatică, clasa a XI-a, curriculum de specialitate (CS)**

- *Filiera vocațională, profilul militar, specializarea matematică-informatică militară*

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 vocațională, profilul militar, specializarea matematică-informatică militară, alocarea orară este:

- clasa a IX-a – 3 ore/săptămână, dintre care o oră pentru studiu teoretic și două ore pentru activități practice;
- clasa a X-a – 3 ore/săptămână, dintre care o oră pentru studiu teoretic și două ore pentru activități practice;
- clasa a XI-a – 3 ore/săptămână, dintre care o oră pentru studiu teoretic și două ore pentru activități practice;
- clasa a XII-a – 3 ore/săptămână, dintre care o oră pentru studiu teoretic și două ore pentru 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** (CDEOS), 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 atestărilor 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 limbajelor de programare Python și 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 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.

Studiul limbajului C++ are o valoare formativă și permite aprofundarea conceptelor fundamentale de programare și a gândirii algoritmice. 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 experiența 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 XI-a	
CS 1.1. Identifică principalele caracteristici ale organizării datelor în cadrul unor modele conceptuale complexe – rețea, ierarhice, obiectuale, pentru a structura și accesa date în vederea prelucrării acestora	
- definirea conceptului de graf, pentru exemplificarea reprezentării conexiunilor dintre posturile de observare ale unei unități militare (neorientat) și a rutelor de transmitere a ordinelor (orientat) - definirea elementelor unui graf neorientat (nod, muchie), pentru exemplificarea reprezentării unei rețele de comunicații militare, unde nodurile reprezintă centrele de comandă, iar muchiile legăturile radio sau cablate dintre acestea - recunoașterea caracteristicilor grafurilor ponderate (orientate sau neorientate), prin completarea unei fișe comparative, pentru exemplificarea reprezentării unei hărți operaționale care are și trasee unidirecționale, respectiv a unei hărți cu trasee bidirecționale, în care costurile arcelor/muchiilor indică distanța sau timpul de deplasare între obiectivele militare - definirea elementelor componente ale unui arbore cu rădăcină (rădăcină, nod, frunză, nivel), pentru exemplificarea reprezentării ierarhiei de comandă a unei structuri militare, unde rădăcina este comandantul unității, nodurile intermediare sunt ofițerii, iar frunzele sunt subunitățile operative - identificarea metodelor de reprezentare a arborilor cu rădăcină, cu referințe ascendente sau referințe descendente, pentru exemplificarea reprezentării conexiunilor din cadrul lanțului de raportare într-o structură militară, de la subordonat la superior (ascendent) și de la comandant la executanți (descendent) - identificarea unor noduri într-un arbore cu rădăcină (nod descendent direct, nod descendent, nod ascendent direct, nod ascendent, nod frate) pentru reprezentarea relațiilor dintre unități, o companie fiind nod descendent direct al batalionului, iar alte companii din același batalion fiind noduri „frați” - identificarea caracteristicilor unei clase, prin exemplul stabilirii membrilor clasei „Rețea”, care încapsulează datele tehnice și metodele de întreținere a unei rețele	
CS 1.2. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea grafurilor	
- precizarea etapelor algoritmului de parcurgere în lățime (BFS), în contextul explorării unei rețele de comunicații militare, unde algoritmul simulează verificarea tuturor legăturilor de pe același nivel - precizarea etapelor algoritmului de parcurgere în adâncime (DFS), în contextul analizării accesibilității nodurilor unei rețele de comunicații, pe care algoritmul le explorează în profunzime - definirea scopului algoritmului lui Prim, indicând tipul de graf pentru care se aplică, în contextul utilizării acestuia pentru analizarea asfaltării unor străzi dintr-un oraș, astfel încât, cu un cost minim de asfaltare, să existe posibilitatea deplasării pe asfalt între oricare două intersecții, eventual trecând pe asfalt și prin alte intersecții	
CS 1.3. Identifică specificul, caracteristicile și etapele unor strategii de tip Backtracking pentru rezolvarea problemelor	
- precizarea caracteristicilor metodei de programare Backtracking, în contextul modelării procesului decizional pentru construirea unui robot, în limita unui buget dat, unde se explorează toate opțiunile posibile de alegere a pieselor necesare, cu testarea condițiilor de continuare și eliminarea ramurilor invalide, revenindu-se la starea anterioară în momentul în care pentru starea curentă au fost explorate toate variantele - identificarea caracteristicilor metodei Backtracking, în contextul generării unor liste de obiective turistice care să fie vizitate succesiv într-un traseu turistic, respectând anumite restricții (durată, conexiuni rutiere etc.) - identificarea principiului de generare sistematică a secvențelor de valori în cadrul metodei Backtracking, în contextul testării succesive a tuturor variantelor de amplasare a unor camere web de observare pe unii dintre stâlpii unei străzi pentru acoperirea completă a acesteia - precizarea caracteristicilor metodei de programare Backtracking, în contextul modelării procesului decizional utilizat într-o misiune militară, unde se explorează toate opțiunile posibile, revenindu-se la starea anterioară când pentru starea curentă au fost explorate toate variantele - identificarea caracteristicilor metodei Backtracking, în contextul utilizării unei liste pentru memorarea etapelor succesive dintr-un plan de recunoaștere, cu posibilitatea de revenire la o etapă parcursă la pasul precedent, pentru construirea unui plan de recunoaștere alternativ - identificarea principiului de generare sistematică a secvențelor de valori în cadrul metodei, în contextul testării succesive a tuturor variantelor de amplasare a unui număr dat de posturi de observare pentru acoperirea optimă a unui traseu liniar	

Clasa a XI-a	
CS 1.4. Identifică principalele elemente ale limbajului de programare utilizate pentru memorarea și prelucrarea datelor organizate în modele complexe - rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice	
- identificarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat sau orientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare - identificarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu liste de adiacență, respectiv a unui graf orientat, cu liste de succesori, în contextul parcurgerii nodurilor acestuia - identificarea instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la fiii oricărui nod	
CS 1.5. Identifică elementele de sintaxă și componentele din definiția claselor, pentru a le utiliza în implementarea algoritmilor în limbaj de programare cu programare orientată pe obiecte	
- identificarea convențiilor de numire a membrilor unei clase (public, privat) și sintaxa prin care sunt stabilite acestea, pentru a recunoaște controlul vizibilității, prin exemplul definirii unei clase „Rețea” unde atributele tehnice sunt private, iar metodele de întreținere sunt publice, pentru a reflecta controlul accesului la date sensibile - identificarea unor elemente de sintaxă pentru definirea unei clase, de exemplu, pentru clasa „Vehicul” cu atribute precum tip, viteză, stoc combustibil și metode pentru actualizarea vitezei sau realimentare - identificarea sintaxei de definire a constructorului unei clase, de exemplu, pentru inițializarea datelor unui utilizator pe o platformă de cursuri online, precum și a drepturilor pe care le are pe aceasta - identificarea convențiilor de numire a membrilor unei clase (public, privat) și sintaxa prin care sunt stabilite acestea, pentru a recunoaște controlul vizibilității, prin exemplul definirii unei clase „Echipament” unde atributele tehnice sunt private, iar metodele de întreținere sunt publice, pentru a reflecta controlul accesului la date sensibile	

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 XI-a	
CS 2.1. Explică principiile de organizare a datelor în cadrul unor modele conceptuale complexe - rețea, ierarhice, obiectuale, pentru a le gestiona eficient	
- explicarea metodelor de reprezentare a unui graf orientat (matrice de adiacență, liste de adiacență), în contextul reprezentării conexiunilor corespunzătoare transiterii ordinilor în cazul punctelor de observare și a posturilor de comandă ale unei unități militare - explicarea rolului elementelor componente ale unei diagrame prin care se reprezintă un graf orientat, respectiv a un graf neorientat în contextul diferenței dintre fluxul unidirecțional de informații (graf orientat, de la comandă spre execuție) și cooperarea bilaterală între unități (graf neorientat) - explicarea relațiilor de descendență și ascendență într-un arbore cu rădăcină, prin exemplificarea structurii de comandă a unei brigăzi, unde relațiile de ascendență indică superiorii ierarhici, iar cele de descendență subunitățile aflate în coordonare - explicarea rolului rădăcinii într-un arbore, prin exemplul identificării comandantului ca element central al unei structuri ierarhice militare, de la care pornesc toate fluxurile decizionale și informaționale majore - explicarea încapsulării, prin exemplul stabilirii membrilor unei clase, „Rețea”, unde se dorește ca datele tehnice să nu fie accesibile direct pentru utilizator, pentru a reflecta controlul asupra datelor sensibile	
CS 2.2. Explică etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea grafurilor	
- explicarea etapelor de funcționare ale algoritmului de parcurgere în lățime (BFS) a grafurilor, prin exemplul identificării rutei optime de comunicare între punctele de distribuție a unui produs (depozite) și punctele de desfacere (magazine), analizând ordinea vizitării nodurilor (depozite și magazine) și a legăturilor dintre acestea - explicarea etapelor de funcționare ale algoritmului lui Dijkstra de determinare a unui drum de cost minim într-un graf, prin exemplul identificării rutei optime de transport între două puncte de distribuție a unui produs (depozite), având în vedere alegerea nodurilor (depozitelor) și actualizarea adecvată a distanțelor rutiere corespunzătoare legăturilor dintre acestea - explicarea etapelor algoritmului lui Prim, prin exemplul construirii unei rețele de comunicații cu un cost total minim, integrând, după caz, pas cu pas, conexiunile cu costul cel mai mic - explicarea etapelor de funcționare ale algoritmului de parcurgere în lățime a grafurilor, prin exemplul identificării rutei optime de comunicare între posturile de observare și centrul de comandă, analizând ordinea vizitării nodurilor și a legăturilor dintre acestea	

Clasa a XI-a	
CS 2.3. Explică etapele unor strategii de tip Backtracking pentru rezolvarea problemelor	
- explicarea modului de generare a tuturor soluțiilor utilizând metoda Backtracking, în contextul procesului de planificare a rutelor unei companii de transport aerian, unde fiecare destinație reprezintă o decizie dependentă de destinația anterioară - explicarea rolului validării pe parcursul determinării soluțiilor cu metoda Backtracking, în contextul verificării condițiilor (acoperire, vizibilitate, siguranță) de amplasare a unor senzori într-o rețea de comunicații, în toate modurile posibile - explicarea procesului de revenire la starea anterioară pentru generarea unei soluții cu metoda Backtracking, prin exemplul epuizării tuturor variantelor de alegere a unui dispozitiv adecvat pentru supravegherea unei anumite zone și necesitatea de a le reconfigura pe cele deja plasate	
- explicarea modului de generare a tuturor soluțiilor utilizând metoda Backtracking, în contextul procesului de planificare a rutelor de patrulare, unde fiecare pas reprezintă o decizie dependentă de pasul anterior - explicarea rolului validării pe parcursul determinării soluțiilor cu metoda Backtracking, în contextul verificării condițiilor tactice (acoperire, vizibilitate, siguranță) de amplasare a unor senzori într-o rețea de comunicații, în toate modurile posibile - explicarea procesului de revenire la starea anterioară pentru generarea unei soluții cu metoda Backtracking, prin exemplul epuizării tuturor variantelor de alegere a unei echipe disponibile pentru supravegherea unei anumite zone și necesitatea de a le reconfigura pe cele deja distribuite	
CS 2.4. Explică rolul principalelor elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele complexe - rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice	
- explicarea rolului instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor dintr-un grup de persoane, în care sunt evidențiate relațiile de prietenie - explicarea modului în care pot fi utilizate instrumentele clasei list pentru reprezentarea în memorie a unui graf orientat, cu liste de predecesori, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de transmitere a unor informații, cu precizarea sursei	
- explicarea rolului instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore binar, astfel încât să permită accesul la fiii oricărui nod, în contextul unui arbore corespunzător unui chestionar, în care fiecare întrebare reprezintă un nod și este urmată, eventual, de o altă întrebare, în funcție de răspunsul dat (corect, incorect)	
CS 2.5. Explică rolul restricțiilor de acces și al componentelor unei clase, pentru a le utiliza în proiectarea claselor și implementarea algoritmilor cu programare orientată pe obiecte	
- explicarea modului în care constructorul inițializează starea unui obiect, pentru a crea corect instanțele, în contextul unui obiect „Concurs” (locație, disciplină, echipă) la crearea acestuia - explicarea modului de utilizare a membrilor clasei pentru a înțelege manipularea obiectelor într-un program, prin exemplul utilizării metodelor din clasa „Rețea” pentru a gestiona dispozitivele din rețea	
- explicarea modului în care constructorul inițializează starea unui obiect, pentru a crea corect instanțele, în contextul unui obiect „Misiune” (locație, durată, echipă) imediat după crearea acestuia - explicarea modului de utilizare a membrilor clasei pentru a înțelege manipularea obiectelor într-un program, prin exemplul utilizării metodelor din clasa „Echipament” pentru a actualiza starea unui dispozitiv din rețea după o misiune (ex. set_status("reparat"))	

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

Clasa a XI-a	
CS 3.1. Utilizează modul de organizare și prelucrare a datelor în cadrul unor modele conceptuale complexe - rețea, ierarhice, obiectuale, pentru a rezolva probleme de gestionare a datelor	
- utilizarea matricei de adiacență pentru a reprezenta un graf neorientat dat prin diagramă, în contextul modelării unei rețele de calculatoare conectate într-un laborator de informatică, evidențiind prezența valorilor nule pe diagonala principală, simetria față de aceasta - utilizarea listelor de succesori pentru a reprezenta un graf orientat dat prin matrice de adiacență, în contextul modelării fluxului de transmitere a datelor într-o rețea de comunicații	
- utilizarea matricei de adiacență pentru a reprezenta un graf orientat dat prin diagramă în contextul modelării fluxului de transmitere a ordinelor între nivelurile de comandă dintr-o structură militară ierarhizată, evidențiind prezența valorilor nule pe diagonala principală, asimetria față de aceasta - aplicarea metodei de reprezentare prin diagramă pentru un arbore dat prin referințe ascendente, în contextul determinării numărului de niveluri ale unui arbore cu rădăcină ce reprezintă o structură organizatorică simplificată, cu niveluri corespunzătoare pentru diferite funcții	

Clasa a XI-a	
- aplicarea metodei de reprezentare cu referințe descendente pentru un arbore dat prin diagramă, în contextul construirii unui arbore cu rădăcină pentru a reprezenta structura organizatorică a unui batalion, cu niveluri corespunzătoare comandamentului, companiilor și plutoanelor - aplicarea încapsulării, prin exemplul stabilirii membrilor unei clase, „Rețea”, unde datele tehnice și metodele de întreținere caracterizează o rețea de calculatoare	
CS 3.2. Aplică algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea grafurilor	
- aplicarea algoritmilor BFS și DFS, pentru parcurgerea grafurilor neorientate, pe baza unei reprezentări prin diagramă a unui exemplu de graf neorientat, în care se marchează nodurile vizitate și se evidențiază particularitățile celor doi algoritmi - aplicarea algoritmilor BFS și DFS, pentru parcurgerea grafurilor neorientate, prin exemplul simulării unei rețele logistice militare în care se verifică accesibilitatea între baze, depozite și puncte de distribuție - aplicarea algoritmului lui Dijkstra pentru a determina distanța minimă între două noduri, într-un graf orientat ponderat dat, prin exemplul identificării rutei optime de aprovizionare de la un depozit către bazele logistice militare, ținând cont de costul deplasării - aplicarea algoritmului lui Prim pentru a determina arborele de acoperire de cost minim într-un graf neorientat ponderat dat, prin exemplul proiectării unei rețele de transport bidirecțional care conectează direct sau indirect toate subunitățile dintr-o structură militară, cu un cost total minim de transport	
CS 3.3. Aplică strategii de tip Backtracking pentru rezolvarea problemelor	
- aplicarea etapelor metodei Backtracking pentru generarea tuturor permutărilor unei mulțimi de patru cifre date, prin exemplul generării tuturor cifrurilor posibile de câte 4 cifre distincte, pentru deblocarea unui instrument de transmitere a informațiilor, știind că ultimele două cifre trebuie să fie pare - aplicarea etapelor metodei Backtracking pentru generarea tuturor combinațiilor unui set de obiecte într-un context cotidian, prin exemplul determinării tuturor posibilităților de a forma grupe de câte un număr fixat de echipamente necesare pentru o misiune - aplicarea etapelor metodei Backtracking pentru generarea sistematică a unor secvențe de valori, determinând eficient ultimele două soluții generate, prin exemplul planificării diferitelor scenarii de deplasare a unor drone care să includă toate punctele de interes	
CS 3.4. Utilizează principalele elemente ale limbajului de programare pentru prelucrarea datelor organizate în modele complexe - rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice	
- utilizarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de pe o hartă, în care sunt evidențiate relațiile de vecinătate dintre țări - utilizarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf orientat, cu liste de succesori, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de transmitere a unor informații, cu precizarea destinației - utilizarea instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la tatăl oricărui nod, în contextul unui arbore asociat unui sistem de fișiere în care rădăcina reprezintă directorul principal, iar fiecare folder (sau fișier) are un singur director părinte	
CS 3.5. Utilizează elementele de sintaxă și componentele din definiția claselor, în implementarea algoritmilor în limbaj de programare cu programare orientată pe obiecte	
- utilizarea elementelor de sintaxă pentru crearea unei clase care utilizează date ale unor clase definite anterior, de exemplu, clasa „Rezolvare” are drept membri obiecte din clasele „Elev”, respectiv „Problema”, asigurând corelarea dintre problemele propuse și elevii care le-au rezolvat - utilizarea elementelor de sintaxă ale limbajului de programare pentru definirea unei clase cu attribute și metode deja stabilite, pentru gestionarea unei situații reale, prin exemplul proiectării unei clase „RaportMisiune” care gestionează datele despre operațiuni militare, de exemplu, codul operațiunii, obiectivul principal, numărul participanților etc. - utilizarea elementelor de sintaxă pentru crearea unei clase cu un constructor și metode pentru gestionarea unei situații reale, prin exemplul implementării unei clase „VehiculBlindat” care include un constructor pentru inițializare (tip, armament, an de fabricație) și metode pentru îmbunătățirea performanțelor (de exemplu, tuning pentru motor), modificare a misiunii curente	

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 XI-a	
CS 4.1. Analizează caracteristicile, modul de prelucrare, avantajele și dezavantajele utilizării unor modele conceptuale complexe - rețea, ierarhice, obiectuale, pentru a alege structura adecvată în rezolvarea unor probleme concrete	
- examinarea gradelor nodurilor unui graf orientat asociat unei rețele de străzi dintr-un oraș pentru a identifica intersecțiile conectate la un număr mare de străzi, în vederea fluidizării traficului din localitate	

Clasa a XI-a	
- <i>analizarea reprezentării prin diagramă a unui graf orientat asociat unei rețele de comunicații, în contextul stabilirii direcțiilor de transmitere a semnalelor</i> - <i>analizarea caracteristicilor unui graf neorientat, conex, în contextul examinării rețelei de patrulare și posturi de observare pentru a verifica dacă toate punctele pot comunica între ele și dacă există trasee circulare de patrulare (cicluri)</i> - <i>analizarea definițiilor echivalente ale unui arbore, în contextul rezolvării unei probleme de natură informatică, în care se verifică organizarea datelor sub formă de arbore, pentru a o alege pe cea mai ușor de implementat</i> - <i>analizarea relațiilor ierarhice în arborii cu rădăcină (înălțime, noduri descendente, noduri ascendente), în contextul comparării mai multor arbori asociați ierarhiilor de comandă corespunzătoare pentru două unități militare pentru a evalua eficiența comunicării și distribuția responsabilităților</i> - <i>analizarea membrilor unei clase pentru a verifica dacă aceasta conține toate datele și metodele necesare pentru a răspunde unei funcționalități, prin exemplul clasei „Rețea”, care cuprinde datele tehnice și metodele de întreținere a unei rețele de calculatoare</i>	
CS 4.2. Analizează comportamentul, aplicabilitatea, avantajele și dezavantajele utilizării unor algoritmi specializați pe clase de probleme pentru prelucrarea grafurilor	
- <i>analizarea metodelor BFS și DFS de parcurgere a grafurilor în vederea utilizării lor pentru verificarea proprietății de conexitate a unui graf neorientat, prin exemplul identificării componentelor izolate dintr-o rețea de comunicații bidirecționale, pentru a determina întreruperi în lanțul de transmitere a datelor</i> - <i>analizarea metodelor BFS versus DFS de parcurgere a unui graf neorientat/orientat în vederea adaptării lor pentru determinarea lanțului/drumului cu lungime minimă între două noduri ale grafului, respectiv pentru determinarea unui drum/lanț între două noduri ale grafului</i> - <i>analizarea rezultatelor parcurgerii în lățime a nodurilor unui graf neorientat, cu interpretarea ordinii nodurilor afișate, prin exemplul comparării ordinii de procesare a informațiilor într-un sistem de comunicații</i> - <i>compararea metodelor de parcurgere a grafurilor neorientate BFS și DFS pentru o alege pe cea adecvată, prin exemplul comparării strategiilor de explorare a unei zone de operații: BFS pentru acoperire rapidă a tuturor punctelor, DFS pentru investigarea completă a unui sector restrâns</i> - <i>analizarea diferențelor dintre metodele de parcurgere BFS și DFS, aplicate unei hărți operaționale militare, pentru parcurgerea cursurilor de acțiune</i> - <i>analizarea complexității algoritmului Dijkstra, prin exemplul evaluării performanței sistemului de planificare a rutelor unui traseu turistic tematic, în care se cunosc distanțele dintre obiective, în funcție de numărul de puncte de destinație și de conexiuni disponibile</i> - <i>analizarea rezultatelor algoritmului lui Prim, în contextul determinării acoperirii de cost minim a unui graf ponderat, prin exemplul construirii unui model grafic care indică legăturile logistice de bază necesare pentru conectarea tuturor sediilor unei firme de transport</i>	
CS 4.3. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de tip Backtracking pentru rezolvarea problemelor	
- <i>analizarea mecanismelor de avans și de revenire în arborele soluțiilor la aplicarea metodei Backtracking, prin exemplul generării tuturor anagramelor cu anumite proprietăți ale unui cuvânt format cu litere distincte (de exemplu, astfel încât literele să nu fie alăturate și în cuvântul dat), pentru evidențierea modului în care algoritmul explorează adecvat posibilitățile și revine pentru a face alte alegeri de litere, în vederea obținerii de noi soluții</i> - <i>analizarea aplicabilității metodei Backtracking, în funcție de numărul de soluții posibile, prin exemplul evaluării timpului de executare pentru planificarea tuturor rutelor aeriene pentru mai multe avioane care zboară simultan</i> - <i>analizarea mecanismelor de avans și de revenire în arborele soluțiilor la aplicarea metodei Backtracking, prin exemplul generării tuturor posibilităților de alocare a resurselor umane și logistice în timpul unei misiuni militare, care să respecte anumite restricții, pentru evidențierea modului în care algoritmul explorează toate posibilitățile și revine pentru a corecta decizii nevalide</i>	
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 complexe - rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice	
- <i>analizarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență sau cu liste de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de identificare a calculatoarelor dintr-o rețea, în care sunt evidențiate relațiile de conexiune directă dintre acestea</i> - <i>analizarea instrumentelor clasei list pentru reprezentarea în memorie, cu liste de predecesori, a unui graf orientat corespunzător unui sistem de transmitere a unor informații, în care vârfurile reprezintă stații de emisie-recepție și pentru fiecare se precizează sursele de la care primește direct informații, în contextul determinării, pentru o stație dată, a tuturor surselor directe sau indirecte de informații</i> - <i>analizarea instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la tatăl oricărui nod, în contextul unui arbore asociat unui sistem ierarhic din cadrul unei organizații în</i>	

Clasa a XI-a
<i>care rădăcina corespunde directorului general, fiecare alt angajat are un singur manager direct, iar în organizație nu există cicluri de subordonare</i>
CS 4.5. Analizează aplicabilitatea, avantajele și dezavantajele utilizării programării orientate pe obiecte, în implementarea algoritmilor în limbaj de programare
- <i>analizarea avantajelor și dezavantajelor utilizării convențiilor de numire a membrilor unei clase pentru a interpreta impactul asupra securității datelor, prin exemplul protejării datelor sensibile dintr-o aplicație de evidență a personalului militar (date de identificare, misiuni clasificate)</i> - <i>analizarea unei aplicații cu mai multe clase pentru a identifica dependențele și interacțiunile dintre obiecte, prin exemplul analizării relațiilor dintre clasele „Misiune”, „Echipament”, „Echipa”, pentru un sistem de planificare operațională</i>

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

Clasa a XI-a
CS 5.1. Argumentează alegerea unor modele conceptuale complexe - rețea, ierarhice, obiectuale și a modurilor de prelucrare a acestora pentru rezolvarea unor probleme informatice concrete
- <i>argumentarea alegerii metodelor de reprezentare a grafurilor neorientate prin matrice de adiacență sau liste de adiacență, în contextul alegerii modului optim de reprezentare a unei rețele de supraveghere video a unei baze militare, având în vedere numărul de noduri ale rețelei și densitatea conexiunilor</i> - <i>evaluarea alegerii modelării printr-un graf orientat a unei situații reale, prin exemplul validării unei diagrame operaționale care descrie legăturile logistice dintre unitățile unei brigăzi</i> - <i>argumentarea alegerii unei metode de reprezentare a arborilor cu rădăcină (referințe descendente, referințe ascendente), în funcție de scopul utilizării, prin exemplul alegerii metodei optime de reprezentare a structurii unei unități militare, în funcție de necesitatea de raportare rapidă sau de detalieri a subunităților</i> - <i>argumentarea alegerii membrilor unei clase, astfel încât aceasta să conțină toate datele și metodele necesare pentru a răspunde unei funcționalități, prin exemplul clasei „Rețea”, care cuprinde datele tehnice și metodele de întreținere a unei rețele de calculatoare</i>
CS 5.2. Evaluează corectitudinea și eficiența soluțiilor cu algoritmi specializați pe clase de probleme pentru prelucrarea grafurilor
- <i>evaluarea menținerii proprietății de conexitate utilizând metodele de parcurgere a unui graf neorientat în contextul unei rețele de comunicații între bazele militare, având în vedere posibilitatea menținerii legăturii între dispozitive în caz de pierdere a unui nod</i> - <i>evaluarea eficienței algoritmului lui Dijkstra pentru determinarea drumurilor de cost minim într-un graf ponderat, prin exemplul determinării diferitelor rute de aprovizionare cu combustibil între un depozit și unitățile militare</i> - <i>evaluarea eficienței algoritmului lui Prim pentru determinarea acoperirii de cost minim într-un graf ponderat, prin exemplul determinării rețelei logistice de transport cu cost minim</i>
CS 5.3. Evaluează corectitudinea și eficiența algoritmilor care utilizează strategii de tip Backtracking pentru rezolvarea problemelor
- <i>evaluarea corectitudinii aplicării metodei Backtracking pentru probleme de generare, prin exemplul determinării modului de parcurgere, într-o croazieră, a unor insule, astfel încât fiecare dintre acestea să fie vizitată o singură dată (cu excepția celei de plecare, la care se și încheie excursia), iar vasul să urmeze doar trasee acceptate între două insule vizitate consecutiv și să aducă turiștii, la final, în punctul de plecare</i> - <i>evaluarea corectitudinii aplicării metodei Backtracking pentru generarea tuturor modurilor de plasare a posturilor de observare, cu anumite restricții privind unele dintre acestea, prin exemplul comparării eficienței explorării exhaustive a variantelor de plasare și verificarea finală a restricțiilor impuse, comparativ cu eliminarea pe parcurs a unor variante necorespunzătoare</i> - <i>evaluarea complexității unui algoritm care folosește metoda Backtracking, prin exemplul estimării timpului necesar pentru explorarea tuturor variantelor de poziționare a unor echipamente într-o rețea de supraveghere</i>
CS 5.4. Evaluează corectitudinea și eficiența aplicațiilor care utilizează elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele complexe - rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice
- <i>evaluarea aplicabilității instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de identificare a unor persoane dintr-un grup, evidențiind relațiile de prietenie dintre acestea</i> - <i>evaluarea instrumentelor clasei list pentru a le alege pe cele adecvate pentru reprezentarea în memorie a unui graf orientat, cu liste de succesori, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de transmitere a unor informații</i>

Clasa a XI-a
- argumentarea utilizării instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la descendenții direcți ai oricărui nod, în contextul unui arbore asociat unui sistem ierarhic din cadrul unei organizații în care rădăcina corespunde directorului general, unii angajați au mai mulți subordonați direcți, iar în organizație nu există cicluri de subordonare
CS 5.5. Evaluează corectitudinea și eficiența programelor care utilizează programare orientată pe obiecte, pentru a rezolva probleme informatice
- evaluarea corectitudinii definirii și utilizării claselor pentru organizarea datelor în contextul implementării unei aplicații de evidență a echipamentelor militare, comparând varianta în care au fost definite clase față de organizarea cu funcții și variabile globale, din perspectiva clarității și modularității codului, cu avantaje pentru întreținerea acestuia și a prevenirii erorilor la actualizarea datelor - evaluarea corectitudinii aplicării convențiilor de numire pentru protecția informațiilor sensibile într-o aplicație reală, prin exemplul unei clase pentru gestionarea datelor specifice unor misiuni militare, comparând accesul liber la date cu utilizarea unor date interne (protejate prin convenții) și a unor metode publice de acces, pentru validarea modificărilor și menținerea consistenței datelor - evaluarea corectitudinii definirii unei clase cu un constructor și metode pentru gestionarea unei situații reale, prin exemplul implementării unei clase „CursOnline” care include un constructor pentru inițializare (denumire, perioadă, număr de ore petrecute online, cost) și metode pentru actualizarea numărului de cursanți sau pentru modificarea perioadei - evaluarea corectitudinii definirii unei clase care are drept membri obiecte ale unor clase definite anterior, de exemplu, clasa „Optiune” are drept membri obiecte ale claselor „Client”, respectiv „Excursie”, asigurând corelarea dintre excursiile propuse și clienții care s-au înscris la acestea

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

Clasa a XI-a
CS 6.1. Proiectează algoritmi personalizați care utilizează modele conceptuale complexe - rețea, ierarhice, obiectuale, pentru organizarea și prelucrarea eficientă a datelor, utilizând algoritmi de bază, în vederea rezolvării unor probleme
- proiectarea unui algoritm care utilizează grafuri neorientate pentru a gestiona datele referitoare la o rețea de comunicații militare între unități, determinând colaboratorii fiecăreia, unitățile cu cei mai mulți colaboratori sau unitățile izolate - proiectarea unui algoritm care utilizează grafuri orientate în contextul modelării căilor de deplasare a autovehiculelor pe străzile dintr-o bază militară, având în vedere că acestea pot fi cu sens unic și că unesc intersecții, determinând străzile cu sens unic și punctele reprezentând extremități ale unor străzi fără ieșire/fundături - proiectarea unui algoritm pentru modelarea unui meniu arborescent, utilizând un arbore cu rădăcină în care fiecare nod corespunde opțiuni pentru o anumită acțiune, cu posibilitatea de a opta și pentru acțiuni subsecvente acesteia - proiectarea unui algoritm care utilizează arbori cu rădăcină, determinând înălțimea arborelui, nodurile frunză, în contextul modelării structurii unei organizații militare, unde înălțimea arborelui reflectă numărul de niveluri ierarhice, iar frunzele reprezintă unitățile operative - proiectarea unui algoritm pentru un model obiectual care încapsulează date și metode, prin exemplul determinării numărului total de exemplare ale publicațiilor dintr-o bibliotecă, în care o publicație este definită ca un model obiectual (titlu, autor, număr de exemplare și metode pentru modificarea numărului de exemplare), iar o bibliotecă este definită ca un model obiectual (listă de publicații, metode pentru adăugarea unei noi publicații)
CS 6.2. Proiectează soluții cu aplicarea personalizată a algoritmilor specializați pe clase de probleme pentru prelucrarea grafurilor
- proiectarea unui algoritm care utilizează metoda de parcurgere în lățime a nodurilor unui graf prin exemplul unui centru de coordonare pentru situații de urgență care trebuie să trimită rapid echipe medicale către toate localitățile, pornind din orașul în care se află centrul, în situația unor intervenții - proiectarea unui algoritm care utilizează metoda de parcurgere în adâncime a nodurilor unui graf în contextul determinării unui traseu dintre două puncte corespunzătoare unor elemente relevante de pe un teren în cadrul unui exercițiu tactic de recunoaștere - proiectarea unui algoritm pentru calcularea costurilor minime de livrare pentru o rețea de depozite, unități operative și rute modelate cu un graf ponderat, utilizând algoritmul lui Dijkstra prin exemplul optimizării transportului logistic militar de la un depozit regional și unități operative, pe baza distanțelor și costurilor de deplasare - proiectarea unui algoritm care utilizează algoritmul lui Prim prin exemplul proiectării unei rețele de transport militar la care să fie conectate prin trasee asfaltate, toate unitățile operative, astfel încât între oricare două puncte să se poată circula pe asfalt, iar lungimea totală a zonelor asfaltate să fie minimă

Clasa a XI-a	
CS 6.3. Proiectează algoritmi cu aplicarea personalizată a strategiilor de tip Backtracking pentru rezolvarea problemelor	
- proiectarea unui algoritm care rezolvă o problemă folosind metoda Backtracking, pentru planificarea poziționării senzorilor de detecție într-o zonă de control, având în vedere anumite restricții - proiectarea unui algoritm folosind metoda Backtracking pentru a determina toate modalitățile de a selecta operațiuni logistice dintr-o mulțime dată, astfel încât acestea să fie din categorii diferite	
CS 6.4. Implementează aplicații care integrează în mod personalizat elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele complexe - rețea, ierarhice, în vederea realizării unor soluții funcționale și performante	
- implementarea unei aplicații care utilizează clasa list pentru reprezentarea în memorie a unui graf orientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de identificare a unor persoane dintr-un grup, evidențiind modul în care o persoană cunoaște sau nu o altă persoană - implementarea unei aplicații care utilizează clasa list pentru reprezentarea în memorie a unui graf neorientat, cu liste de muchii, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de comunicare între prieteni - implementarea unei aplicații care utilizează instrumentele clasei list pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la ascendentul direct al oricărui nod, în contextul unui arbore asociat unui sistem ierarhic din cadrul unei organizații în care rădăcina corespunde directorului general, iar fiecare alt angajat are un singur manager	
CS 6.5. Implementează aplicații în limbaj de programare care integrează clase personalizate, pentru a modulariza și organiza eficient codul în cadrul soluțiilor informatice	
- implementarea unei aplicații de gestionare a bibliotecii militare de unitate, cu clase pentru cărți, utilizatori și evidența împrumuturilor - implementarea unei aplicații interne de logistică pentru gestionarea comenzilor de echipamente și uniforme militare, cu clase pentru echipamente, uniforme și depozit - implementarea unui simulator tactic pentru antrenarea în vederea îndeplinirii obiectivelor misiunii, având clase pentru „Echipament”, „Vehicul” și „Soldat”	

CONȚINUTURI ALE ÎNVĂȚĂRII

Clasa a XI-a

Domenii de conținut	Conținuturi
1. Organizarea conceptuală a datelor	1.1. Modelul conceptual rețea - graf - concepte de bază și caracteristici: graf neorientat și orientat, nod și vârf, muchie și arc, adiacență, incidență, grad, grad interior, grad exterior, lanț și drum, lanț elementar și drum elementar, lanț simplu și drum simplu, lanț compus și drum compus, ciclu și circuit, ciclu elementar și circuit elementar; - metode de reprezentare a unui graf: diagramă, matrice de adiacență, liste de adiacență, liste de succesori, liste de predecesori, liste de muchii, liste de arce; - concepte de bază și caracteristici ale unor grafuri asociate unui graf: subgraf, graf parțial; - concepte de bază, caracteristici și proprietăți ale unor tipuri de grafuri: graf complet, graf conex, graf tare conex (matricea drumurilor), graf ponderat (matricea costurilor); - repere pentru aplicarea algoritmilor de bază în prelucrarea grafurilor. 1.2. Model conceptual ierarhic - arbore - caracteristici și definiții echivalente ale unui arbore, respectiv arbore cu rădăcină; - concepte de bază într-un arbore cu rădăcină și caracteristici ale acestora: rădăcină, nivel al unui nod, nod frunză, nod descendent direct/fiu, nod descendent, nod ascendent direct/tată, nod ascendent, nod frate; - metode de reprezentare a unui arbore cu rădăcină: diagramă, referințe ascendente, referințe descendente; - repere pentru aplicarea algoritmilor de bază pentru prelucrarea arborilor. 1.3. Model conceptual obiectual – clase și obiecte - concepte și caracteristici: clasă, obiect; - repere pentru aplicarea algoritmilor de bază cu date organizate obiectual.
2. Strategii de rezolvare a problemelor	2.1. Metoda Backtracking - caracteristici ale metodei Backtracking; - repere pentru aplicarea metodei de generare sistematică a unor secvențe de valori. 2.2. Algoritmi specifici pentru prelucrarea grafurilor - caracteristici, etape ale metodelor de parcurgere în lățime (Breadth First Search - BFS), respectiv în adâncime (Depth First Search – DFS) și repere pentru aplicarea acestora; - repere pentru utilizarea metodelor de parcurgere pentru verificarea proprietății de conexitate a unui graf neorientat și determinarea componentelor conexe; - repere pentru utilizarea metodelor de parcurgere pentru determinarea matricei drumurilor/lanțurilor unui graf; - repere pentru utilizarea matricei drumurilor pentru verificarea proprietății de tare conexitate și determinarea componentelor tare conexe ale unui graf orientat; - caracteristici, etape ale algoritmului lui Dijkstra pentru determinarea drumurilor de cost minim într-un graf ponderat și repere pentru aplicarea acestuia; - caracteristici, etape ale algoritmului lui Prim pentru determinarea arborelui de acoperire de cost minim într-un graf ponderat și repere pentru aplicarea acestuia.
3. Memorarea datelor și organizarea codului în limbaj de programare	3.1. Programare orientată pe obiecte - sintaxa definirii unei clase, niveluri de acces la membrii clasei, constructor, destructor/finalizer; - repere pentru definirea unor clase simple. 3.2. Paradigme de bază în programare - caracteristici, tipuri de paradigme (de exemplu, procedurală, obiectuală, declarativă); - concepte, funcționalități, avantaje și dezavantaje ale altor limbaje de programare (de exemplu, C, C++, Java, JavaScript, C#, J#).

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 măsoară? 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.

Un context de învățare poate viza simultan mai multe competențe, mai ales atunci când acestea sunt interconectate. De exemplu, pentru rezolvarea unei probleme cu ajutorul calculatorului se vizează, în cadrul aceleiași activități de învățare, atât competențe de nivel cognitiv scăzut (identificarea principalelor caracteristici ale modelelor conceptuale și operaționale implicate, explicarea principiilor care stau la baza acestora, în contextul particular al problemei, utilizarea lor în rezolvare) dar și competențe de nivel cognitiv ridicat (analizarea caracteristicilor modelelor conceptuale și operaționale specifice problemei, evaluarea corectitudinii și eficienței soluțiilor propuse și elaborarea algoritmului personalizat).

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, algoritimizarea, 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 algoritmi 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 algoritmi 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 metode/algoritmi 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, urmărirea 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 XI-a, specializarea matematică-informatică militară:

1. *Strategii de rezolvare a problemelor. Metoda Backtracking*
2. *Organizarea conceptuală a datelor. Model conceptual rețea – graf*
3. *Strategii de rezolvare a problemelor. Algoritmi specifici pentru prelucrarea grafurilor*
4. *Organizarea conceptuală a datelor. Model conceptual ierarhic – arbore*
5. *Strategii de rezolvare a problemelor. Algoritmi specifici pentru prelucrarea arborilor*
6. *Organizarea conceptuală a datelor. Model conceptual obiectual*
7. *Memorarea datelor și organizarea codului în limbaj de programare. Programare orientată pe obiecte*
8. *Memorarea datelor și organizarea codului în limbaj de programare. Paradigme de bază în programare*

Pentru predarea metodei Backtracking se recomandă precizarea unor repere avute în vedere pentru rezolvarea problemelor (de exemplu, ce este o soluție dintre cele cerute, ce reprezintă un element al acesteia, care sunt condițiile de validare a unei valori propuse, care este condiția de obținere a unei soluții complete) și exemplificarea acestora pentru cazuri concrete. Se recomandă abordarea unor rezolvări cu aplicarea generării de elemente combinatoriale (de exemplu, permutări, aranjamente, combinații, produs cartezian), eventual cu restricții privind relațiile dintre elemente, probleme clasice în care o soluție se poate reprezenta ca o succesiune de perechi, în care poziția are o anumită semnificație (de exemplu, problema reginelor, în care o soluție este o succesiune de poziții pe tabla de șah, perechi linie-coloană, iar numărul de ordine în cadrul soluției reprezintă linia și valoarea corespunzătoare reprezintă coloana, problema colorării unei hărți, în care o soluție este o succesiune de perechi țară-culoare, iar numărul de ordine în cadrul soluției reprezintă țara și valoarea corespunzătoare reprezintă culoarea), probleme în care sunt preluate valori parțiale calculate la pașii anteriori (de exemplu, problema partițiilor unui număr, problema plății unei sume cu anumite tipuri de monede, problema partițiilor unei mulțimi), probleme de optim (de exemplu, problema unui comis-voiajor) ca model, apoi realizarea unor probleme asemănătoare care presupun adaptări, reinterprețări, extinderi.

Se recomandă ca în cadrul temelor Model conceptual rețea – graf și Metode de prelucrare a grafurilor reprezentările grafurilor (diagramă, matrice de adiacență, liste) să fie introduse succint, cu exemple intuitive (hartă, rețea de prieteni), iar accentul să fie pus ulterior pe utilizarea algoritmilor specifici, în mod structurat, urmând etape corespunzătoare pentru elaborarea programelor. Programa este flexibilă din punctul de vedere a ordinii de predare, astfel încât se pot preda, de exemplu, în paralel grafurile neorientate și orientate, apoi arborii, pentru a elimina redundanțele și a pune în evidență diferențele dintre acestea, sau se pot preda succesiv, pentru a fixa noțiunile specifice, de exemplu, grafuri neorientate, grafuri orientate, arbori sau grafuri neorientate, arbori, grafuri orientate. Astfel, pentru grafuri, se pot preda concepte de bază (definiții ale grafurilor neorientate/orientate, concepte ca adiacență, grade, lanț/drum, ciclu/circuit), metode de reprezentare și aplicații cu algoritmi de bază, de exemplu, pentru determinarea gradului unui nod/vârf, a nodurilor izolate, verificarea proprietății de lanț/drum pentru un șir de valori. După aceea, se pot prezenta metodele de parcurgere a grafurilor, grafuri asociate și tipuri de grafuri: grafuri complete, grafuri conexe (definiții și utilizarea metodelor de parcurgere în aplicații), grafuri tare conexe (definiții, algoritmi specializați pentru determinarea matricei drumurilor și utilizarea lor în aplicații), grafuri ponderate (definiții, algoritmi specializați pentru determinarea unor lanțuri/drumuri de cost minim și utilizarea acestora în aplicații). La liceu nu se studiază grafuri cu bucle, grafuri infinite sau multigrafuri, considerându-se că oricare arc/muchie are extremități distincte și oricare două arce/muchii diferă prin cel puțin una dintre extremități. De remarcat că în lucrările de specialitate, algoritmul Roy-Floyd se regăsește și cu denumirea Floyd-Warshall.

În cadrul temei Model conceptual ierarhic – arbore, se recomandă să se pornească de la modele vizuale reale (arbore genealogic, ierarhie școlară, meniuri de site), iar după aceea să fie introduse concepte formale. Implementarea arborilor binari în Python se face utilizând clasa list, fiecare înlănțuire cu un nod ascendent/descendent fiind dată prin poziția la care se află acesta.

Pentru tema Programare orientată pe obiecte, se recomandă evidențierea faptului că programarea orientată pe obiecte nu înseamnă doar „alt tip de sintaxă”, ci un mod de gândire în care problemele se modelează folosind entități (obiecte) și tipuri (clase), iar accentul se mută de la „ce face programul pas cu pas” la „cine (obiectul) face acțiunea”.

Proiectele interdisciplinare și învățarea pe baza unor aplicații care să conecteze teoria cu lumea reală (de exemplu, pentru grafuri, modelarea rețelelor sociale, a rutelor între orașe, pentru arbori, modelarea unor ierarhii organizaționale, a arborelui genealogic) 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.

Pentru tema Paradigme de bază în programare se vor prezenta succint conținuturile din programă, având în vedere diferențe și asemănări, prezentarea unui exemplu de cod în Python și în alte limbaje pentru principalele aspecte evidențiate, punând în valoare facilitățile limbajelor/paradigmelor.

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,

Nume și prenume	Grad didactic/Titlu științific	Instituție de apartenență, localitate, județ
		Facultatea de Automatică și Calculatoare
NIȚU Alina	profesor, gradul didactic I	Liceul Teoretic „Ovidius”, Constanța, județul Constanța
NURLA Aidan	profesor, gradul didactic I	Colegiul Național Militar „Alexandru Ioan Cuza”, Constanța, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forțelor 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 Național Militar „Dimitrie Cantemir”, Breaza, județul Prahova
TEGLAȘ Maria	profesor, gradul didactic II	Colegiul Național 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

Anexa nr. 7

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. 7
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 (Ordinul 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 limbajelor de programare Python și 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 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.

Studiul limbajului C++ are o valoare formativă și permite aprofundarea conceptelor fundamentale de programare și a gândirii algoritmice. 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 algoritmi 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 - 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 - 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>i</i>, pe poziția <i>j</i>, reprezintă media elevului <i>i</i> la disciplina <i>j</i>	
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 - enumerarea etapelor metodei căutării binare în contextul localizării unui titlu într-o listă ordonată de cărți dintr-o bibliotecă digitală - 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 - 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 - 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.) - 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 - 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 - 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 - 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>i</i>, pe poziția <i>j</i>, este 1, dacă țările <i>i</i> și <i>j</i> sunt vecine 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

Clasa a X-a	
- explicarea modului în care este prelucrat un șir de caractere, în contextul funcționării unui sistem de verificare a complexității parolilor, care analizează fiecare simbol pentru a identifica litere mari, litere mici, cifre și caractere speciale - 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 - 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 - 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) - 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 - 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 $3+3$) - 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()) - 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	

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 - 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) - 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 - 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	

Clasa a X-a	
- 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 - 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 - 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 - 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 - 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) - aplicarea metodei de sortare prin interclasare pentru obținerea unei liste ordonate pornind de la o listă neordonată formată din 7 numere date - 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 - 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()) - 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()) - 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	

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 - 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) - 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 - 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 - 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	

Clasa a X-a	
- 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 - 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 - 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 - 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 - 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 - 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	

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 - 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 - 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) - 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 - 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 - 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	

Clasa a X-a	
- 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 - evaluarea obținerii soluției optime pentru probleme rezolvate cu metoda Greedy (de exemplu, organizarea unui program de întâlniri fără suprapuneri), prin justificarea corectitudinii condițiilor alese - 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, $\text{card}(A)=n$ și $\text{card}(B) \geq n$), 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 - argumentarea alegerii unei metode din clasa str pentru o sarcină practică, de exemplu, căutarea unor vocale într-un text dat - 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 - 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 - 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 - 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 - 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 - 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	
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 - 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')	

Clasa a X-a

- 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)
- 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

CONȚINUTURI ALE ÎNVĂȚĂRII**Clasa a X-a**

Domenii de conținut	Conținuturi
1. Organizarea conceptuală a datelor	1.1. Modelul conceptual liniar – șir de caractere - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.2. Modelul conceptual mixt (de exemplu, liste de liste, liste de șiruri de caractere) - 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 - caracteristici ale metodei căutării binare și repere pentru aplicarea acesteia; - caracteristici ale metodei interclasării și repere pentru aplicarea acesteia. 2.2. Metoda Greedy - caracteristici ale metodei Greedy; - 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 - caracteristici; - operații și operatori specifici: apartenență, concatenare, multiplicare, acces la un caracter sau subșir/secvență de caractere, comparare lexicografică; - 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; - repere pentru identificarea și utilizarea altor metode și funcții adecvate pentru prelucrarea șirurilor de caractere, în funcție de nevoile proprii. 3.2. Elemente de limbaj pentru operarea cu date organizate în modele mixte - caracteristici, acces la un element în Python.

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 măsoară? 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.

Un context de învățare poate viza simultan mai multe competențe, mai ales atunci când acestea sunt interconectate. De exemplu, pentru rezolvarea unei probleme cu ajutorul calculatorului se vizează, în cadrul aceleiași activități de învățare, atât competențe de nivel cognitiv scăzut (identificarea principalelor caracteristici ale modelelor conceptuale și operaționale implicate, explicarea

principiilor care stau la baza acestora, în contextul particular al problemei, utilizarea lor în rezolvare) dar și competențe de nivel cognitiv ridicat (analizarea caracteristicilor modelelor conceptuale și operaționale specifice problemei, evaluarea corectitudinii și eficienței soluțiilor propuse și elaborarea algoritmului personalizat).

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, algoritimizarea, 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 algoritmi 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 algoritmi de bază (pentru numărare, sumă, produs, minim, maxim, prima sau ultima valoare cu o anumită proprietate, verificarea unor proprietăți) studiate 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 metode/algoritmi 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.magicsschool.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 Național Militar „Alexandru Ioan Cuza”, Constanța, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forțelor 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 Național Militar „Dimitrie Cantemir”, Breaza, județul Prahova
TEGLAȘ Maria	profesor, gradul didactic II	Colegiul Național 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

Anexa nr. 8

Programa școlară
pentru disciplina

Informatică

Clasa a IX-a

Curriculum de specialitate (CS)

*Filiera teoretică, profilul real, specializarea matematică-informatică,
clase cu predarea disciplinei informatică în regim intensiv*

Învățământ liceal

- 2026 –

Anexa nr. 8**Programa școlară pentru disciplina *Informatică*, clasa a IX-a, curriculum de specialitate (CS)**

- *Filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei informatică în regim intensiv*

NOTĂ DE PREZENTARE**Statutul disciplinei**

Disciplina *informatică*, studiată în ciclul liceal, este o continuare a disciplinei *informatică și TIC*, studiată la gimnaziu, în trunchiul comun. În ciclul liceal, aceasta își consolidează și extinde domeniile de competență, aprofundând aspectele 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 predă 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 matematică - informatică, clase cu predarea disciplinei *informatică* în regim intensiv, conform O.M.E.C. nr. 6873/2025 alocarea orară este:

- clasa a IX-a – 4 ore/săptămână, dintre care două ore pentru studiu teoretic și două ore pentru activități practice;
- clasa a X-a – 4 ore/săptămână, dintre care două ore pentru studiu teoretic și două ore pentru activități practice;
- clasa a XI-a – 7 ore/săptămână, dintre care patru ore pentru studiu teoretic și trei ore pentru activități practice;
- clasa a XII-a – 7 ore/săptămână, dintre care patru ore pentru studiu teoretic și trei ore pentru activități practice.

Activitățile practice sunt desfășurate obligatoriu în laboratorul de informatică, iar în baza Ordinului ministrului educației și cercetării nr. 6873/2025, pentru filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei *informatică* în regim intensiv, activitățile practice se desfășoară cu colectivul de elevi organizat pe grupe.

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_OM_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, 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** (CDEOS), 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 atestatorilor 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 limbajelor de programare Python și 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 arte digitale.

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 de programare Python 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.

Studiul limbajului C++ are o valoare formativă și permite aprofundarea conceptelor fundamentale de programare și a gândirii algoritmice. Limbajul 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 experiența 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 (CG, CS, 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 IX-a
CS 1.1. Identifică principalele caracteristici ale organizării datelor în cadrul unor modele conceptuale fundamentale - date simple sau liste, pentru a structura și accesa date în vederea prelucrării acestora
- recunoașterea caracteristicilor unei liste evidențiind modul în care se succed mașinile care sunt oprite la semafor pe o stradă cu sens unic, faptul că fiecare mașină, cu excepția primei și ultimei mașini, are imediat înaintea ei și imediat după ea câte o altă mașină - enumerarea reperelor pentru parcurgerea unei liste în vederea identificării perechilor de elemente aflate pe poziții consecutive și care au aceleași caracteristici - recunoașterea caracteristicilor unei liste cu acces direct prin nominalizarea celui de al cincilea elev din catalog, fără a fi necesară parcurgerea tuturor datelor elevilor care îl preced în catalog - recunoașterea caracteristicilor unei stive și ale unei cozi, din punctul de vedere al modului în care se accesează și gestionează elementele, în contextul exemplificării stivuirii unor farfurii, respectiv așezării unor persoane la un rând, pentru a cumpăra bilete la cinema
CS 1.2. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a fi putea utiliza în prelucrarea algoritmică a numerelor, sortarea sau generarea sistematică a unor secvențe de valori
- enumerarea etapelor algoritmului lui Euclid pentru determinarea celui mai mare divizor comun cu scopul determinării dimensiunii maxime a unei plăci de gresie de formă pătrată folosită pentru acoperirea completă, cu plăci întregi, a podelei unei săli de dimensiuni date - enumerarea etapelor algoritmului de transformare în baza 2 a unui număr scris în baza 10 în contextul reprezentării în memorie a unui număr natural (de exemplu, codul ASCII asociat unei litere) - reamintirea secvenței de operații necesare pentru a adăuga o cifră la dreapta celorlalte cifre ale unui număr, în contextul determinării valorii rezultate în urma eliminării tuturor cifrelor impare ale unui număr, evidențiind necesitatea păstrării ordinii inițiale a cifrelor pare rămase și a cifrelor nule de la finalul numărului - recunoașterea caracteristicilor algoritmului Ciurul lui Eratostene utilizat pentru determinarea numerelor prime până la o limită dată, prin enumerarea etapelor principale (inițializarea listei numerelor, eliminarea multiplilor fiecărui număr prim, obținerea listei finale) și identificarea modului de reprezentare a datelor în listă - reamintirea regulii de obținere a unui termen al șirului Fibonacci, pe baza celor doi termeni anteriori, în contextul unui proiect de cercetare având ca temă șirul lui Fibonacci în natură - identificarea caracteristicilor metodei de sortare prin selecția minimului aplicată scenariului de aranjare a finaliștilor unui concurs de talente în ordinea crescătoare a scorului obținut - recunoașterea caracteristicilor metodei bulului (compararea elementelor adiacente, interschimbarea celor aflate în ordine necorespunzătoare) aplicată unui context real, precum ordonarea înălțimilor elevilor unei clase
CS 1.3. Identifică specificul, caracteristicile și etapele unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor
- identificarea datelor cu care lucrează algoritmii, pe baza caracteristicilor acestora (date de intrare, date ieșire, date de manevră) - conceperea unei diagrame care ilustrează etapele de elaborare a unui program - enumerarea etapelor de elaborare a unui program, precizând rolul fiecăreia - asocierea unor termeni precum modul, gândire computațională, proiectare cu rolul lor în rezolvarea unei probleme informatice
CS 1.4. Identifică principalele elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program, în vederea rezolvării eficiente a problemelor informatice
- recunoașterea operatorilor specifici clasei list din Python ([], in, not in, +, *, ==, !=, <, >), în contextul recapitulării modalităților de lucru cu structuri de date, prin completarea unui tabel care asociază fiecare operator cu rolul său - descrierea unor metode suplimentare ale clasei list din Python (extend(), reverse(), clear(), remove(), sort(), copy()), în contextul recapitulării instrumentelor de prelucrare a colecțiilor de date, prin completarea unui tabel care asociază fiecare metodă cu acțiunea sa principală asupra listei - recunoașterea operatorului specific pentru accesarea unui element al unui tablou unidimensional în C++, prin identificarea sintaxei corecte a operatorului [] și indicarea unui exemplu concret de utilizare, precum accesarea temperaturii din a treia zi a săptămânii dintr-un tablou de temperaturi zilnice

Clasa a IX-a
- enumerarea etapelor în lucrul cu un fișier text și a metodelor corespunzătoare în Python: <code>open()</code>, <code>read()</code>, <code>write()</code> și <code>close()</code> - identificarea pe un exemplu de program a subprogramelor asociate butoanelor sau evenimentelor de tip "apăsarea unei taste"
CS 1.5. Identifică elementele de sintaxă și componentele din definiția și apelul subprogramelor și subprogramelor predefinite pentru operații uzuale, pentru a le utiliza în implementarea algoritmilor în limbaj de programare
- descrierea structurii de bază a unui subprogram în Python (definire cu <code>def</code>, parametri, instrucțiuni, apel) în contextul recapitulării modului de organizare a codului într-un proiect existent, prin completarea unui șablon de cod parțial - recunoașterea funcțiilor predefinite Python pentru calcule matematice uzuale (<code>abs()</code>, <code>round()</code>, <code>int()</code>, <code>sqrt()</code>), într-un program dat, de rezolvare a unei probleme de calcul numeric - identificarea funcțiilor predefinite Python utilizate pentru colecții (<code>len()</code>, <code>min()</code>, <code>max()</code>, <code>sum()</code>), în contextul recapitulării elementelor de bază privind manipularea listelor numerice, prin completarea unui tabel care asociază fiecare funcție cu rezultatul obținut pentru o listă dată - recunoașterea elementelor de sintaxă utilizate în definirea și apelul unui subprogram în C++, prin enumerarea părților componente ale unei funcții (tipul de retur, numele, parametrii, corpul funcției) și identificarea unui exemplu simplu din viața reală, precum o funcție care calculează media aritmetică a două numere introduse de utilizator

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 IX-a
CS 2.1. Explică principiile de organizare a datelor în cadrul unor modele conceptuale fundamentale - date simple sau liste, pentru a le gestiona eficient
- explicarea modului în care principiile de organizare ale unei stive și ale unei cozi reflectă situații din viața reală, exemplificate prin stiva de farfurii dintr-o bucătărie (LIFO) și coada de persoane la ghișeu (FIFO), pentru a evidenția diferențele de acces - explicarea rolului unei liste de frecvențe în organizarea și interpretarea datelor dintr-o situație reală, exemplificată printr-o listă care centralizează voturile obținute de mai mulți candidați, interpretând semnificația valorilor înregistrate în listă ca frecvențe ale opțiunilor exprimate - explicarea modului în care parcurgerea secvențială a datelor permite extragerea de informații esențiale dintr-o colecție organizată liniar, exemplificând prin numărarea persoanelor care au participat la un eveniment, pe baza unui criteriu dat - explicarea necesității memorării datelor într-o listă pentru a permite prelucrări ulterioare, exemplificând prin înregistrarea zilnică a numărului de clienți ai unui magazin într-o lună pentru a identifica zilele cu numărul minim de clienți
CS 2.2. Explică etapele algoritmilor specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea algoritmică a numerelor, sortarea sau generarea sistematică a unor secvențe de valori
- explicarea procesului de formare a unui număr nou prin adăugarea repetată a câte unei cifre în stânga celor existente, evidențiind înmulțirea cifrei adăugate cu o variabilă, pe baza unui algoritm dat - explicarea rolului ridicării la pătrat în calculul unei puteri cu exponent par, pas utilizat în exponențierea rapidă - explicarea modului de formare a unei liste de valori ai cărei termeni respectă o anumită regulă (de exemplu, lista temperaturilor medii anuale în procesul de încălzire globală știind temperaturile din ani consecutivi) - explicarea aplicabilității proporției de aur în știință, artă și în natură, pe baza materialelor de documentare primite, privind șirul lui Fibonacci - explicarea procesului de aranjare în ordine crescătoare a șapte comenzi online în funcție de valoarea comenzilor, folosind selecția minimului, cu reprezentarea ordinii comenzilor după fiecare parcurgere
CS 2.3. Explică etapele unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor
- explicarea etapelor a două strategii de rezolvare prin proiectarea modulară a algoritmilor pentru aceeași problemă, subliniind diferențele de organizare a modulelor și impactul acestora asupra clarității și eficienței soluției - explicarea rolului și importanței fiecărei etape din procesul de dezvoltare a unui program - explicarea rolului fiecărui modul în cadrul unei aplicații cunoscute, pentru realizarea obiectivelor acesteia
CS 2.4. Explică rolul principalelor elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program în vederea rezolvării eficiente a problemelor informatice
- explicarea modului în care caracteristicile clasei list (mutabilitate, acces prin index, ordinea inserării) influențează manipularea datelor, în contextul unei aplicații de gestionare a notelor elevilor, prin descrierea pașilor de adăugare, modificare și ștergere a elementelor - explicarea modului de funcționare a metodelor <code>append()</code>, <code>insert()</code> și <code>pop()</code> în contextul unei aplicații de gestionare a unei liste de participanți la un eveniment, prin justificarea efectelor fiecărei metode asupra structurii listei

Clasa a IX-a	
- interpretarea modului în care operatorul <code>[]</code> permite accesarea unui element dintr-un tablou unidimensional în C++, exemplificând prin gestionarea scorurilor obținute de un jucător în 10 runde de joc, pentru a explica cum indicele stabilește legătura dintre poziția elementului și valoarea corespunzătoare din tablou - explicarea mecanismelor de citire și scriere într-un fișier și a noțiunii de "sfârșit de fișier" - explicarea rolului principalelor obiecte grafice într-o aplicație mai complexă, dată, care utilizează cât mai multe obiecte grafice (fereastră, etichetă, buton, casetă de text, listbox, messagebox, meniu)	
CS 2.5. Explică mecanismul de executare a subprogramelor și subprogramelor predefinite pentru operații uzuale, pentru a le utiliza în implementarea algoritmilor în limbaj de programare	
- explicarea rolului subprogramelor într-un proiect de organizare modulară a unui algoritm dat, care determină câte numere dintr-un șir de valori au suma cifrelor un număr prim, prin descrierea relației dintre antet, parametri și corpul fiecărui subprogram - explicarea modului de funcționare a apelului unui subprogram în contextul unei aplicații de calcul al costului total al cumpărăturilor, prin descrierea legăturii dintre parametri, valoarea returnată și afișarea rezultatului - explicarea modului de funcționare a subprogramelor predefinite pentru conversii și rotunjiri, în contextul unui exemplu practic de prelucrare a notelor școlare, prin precizarea diferenței dintre <code>int()</code> și <code>round()</code> - explicarea modului de funcționare a programelor ce utilizează funcțiile <code>len()</code>, <code>min()</code>, <code>max()</code> și <code>sum()</code> în contextul unei aplicații de gestiune a notelor elevilor, prin argumentarea legăturii dintre fiecare funcție și informația pe care o furnizează despre colecție - explicarea modului în care sintaxa de definire și apel a unui subprogram C++ permite organizarea clară a codului într-o aplicație care determină câte numere palindrom se află într-un șir de valori, prin exemplificarea rolului parametrilor și al valorii returnate în procesul de comunicare dintre subprogramele definite și funcția <code>main</code> - explicarea modului în care funcția <code>floor()</code> din headerul <code><cmath></code> din biblioteca standard C++ este utilizată pentru determinarea părții întregi a unei valori zecimale, prin exemplificarea în contextul calculării numărului întreg de pachete necesare pentru livrarea unor produse, atunci când rezultatul unei împărțiri indică o valoare fracționară	

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

Clasa a IX-a	
CS 3.1. Utilizează modul de organizare și prelucrare a datelor în cadrul unor modele conceptuale fundamentale - date simple sau liste, pentru a rezolva probleme de gestionare a datelor	
- aplicarea modului de funcționare al stivei și al cozii în organizarea sarcinilor zilnice, prin simularea unei liste de activități unde ultima activitate adăugată este prima activitate rezolvată (stivă) sau prima activitate adăugată este prima activitate executată (coadă), evidențiind relevanța alegerii tipului de acces în funcție de context - aplicarea conceptului de listă de frecvențe în analiza răspunsurilor dintr-un chestionar de satisfacție, prin numărarea aparițiilor fiecărei variante de răspuns (1–5 stele) și organizarea rezultatelor sub formă de listă pentru identificarea tendințelor generale - aplicarea principiilor parcurgerii liniare pentru folosirea valorilor care respectă o anumită condiție într-un context real, cum ar fi verificarea listelor de plăți lunare pentru determinarea facturilor restante, prin procesarea secvențială a fiecărui element fără memorarea rezultatelor intermediare - aplicarea conceptului de parcurgere liniară cu memorare prin stocarea notelor obținute de elevii dintr-o clasă la un test, utilizând lista pentru determinarea elevilor care au notele mai mari decât media clasei	
CS 3.2. Aplică algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea algoritmică a numerelor, sortarea sau generarea sistematică a unor secvențe de valori	
- simularea procesului de obținere a divizorilor unui număr natural dat utilizând reperele pentru parcurgerea acestora - implementarea unor algoritmi de divizibilitate dați pentru a verifica dacă un număr natural este perfect, determinând suma divizorilor acestuia - reprezentarea în pseudocod a algoritmului lui Euclid pentru determinarea celui mare divizor comun a două numere naturale nenule citite, pentru utilizarea acestuia în contextul simplificării unei fracții - aplicarea algoritmului de obținere a numerelor prime până la o valoare dată, cu ajutorul ciurului lui Eratostene, ilustrând evoluția listei de valori eliminate/rămase după fiecare etapă de filtrare - aplicarea algoritmului de exponențiere rapidă pentru calcularea cheilor publice și private într-un sistem simplificat de criptare de tip RSA, prin implementarea algoritmului într-un limbaj de programare - aplicarea algoritmului de generare a elementelor unei liste ai cărei termeni respectă o regulă dată, în contextul determinării numărului de ani după care soldul contului bancar, având o sumă inițială și o dobândă anuală, depășește o valoare dată	

Clasa a IX-a	
- implementarea algoritmului de sortare cu listă de frecvențe pentru ordonarea notelor la testul de la informatică al unei clase de elevi (note întregi între 1 și 10), cu construirea listei de frecvențe prin numărarea frecvenței fiecărei note și afișarea notelor sortate prin parcurgerea listei - aplicarea metodei bulelor pentru ordonarea tuturor secvențelor de numere pozitive dintr-o listă (de exemplu, o listă de temperaturi înregistrate într-un oraș), prin implementarea algoritmului într-un limbaj de programare	
CS 3.3. Aplică strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor	
- descompunerea unei probleme, având descrierea rezolvării acesteia, în subprobleme (module), indicând etapele de lucru pentru fiecare - utilizarea principiilor de proiectare modulară pentru o aplicație simplă (de exemplu, gestionarea unor cheltuieli și încasări lunare), atribuind câte un modul fiecărui membru al unei echipe - aplicarea unui model de proiectare modulară în realizarea unei soluții, în vederea efectuării etapelor de analiză, proiectare, testare în cadrul fiecărui modul	
CS 3.4. Utilizează principalele elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program, în vederea rezolvării eficiente a problemelor informatice	
- aplicarea operatorilor de concatenare (+) și multiplicare (*) în rezolvarea unei situații practice de gestionare a comenzilor din două depozite, prin combinarea listelor de produse și dublarea unui stoc existent - aplicarea metodelor count() și index() în rezolvarea unei situații practice de numărare a produselor identice și de identificare a poziției unui articol într-o listă de stocuri, prin scrierea unui program Python simplu - aplicarea metodelor remove() și clear() pentru prelucrarea datelor într-o aplicație de gestionare a unui coș de cumpărături, prin scrierea unui program Python simplu care elimină produsele selectate de utilizator și golește lista după finalizarea comenzii - utilizarea sintaxei de declarare a variabilelor de tip tablou unidimensional în C++ prin modelarea stocului unui magazin cu 100 de produse, stabilind tipul de date potrivit, dimensiunea tabloului și denumirea sugestivă a variabilei, astfel încât fiecare element al tabloului să reprezinte cantitatea disponibilă pentru un produs - completarea unui program "lacunar" care citește o valoare întreagă dintr-un fișier text pentru a scrie diverse rezultate obținute tot într-un fișier text (de exemplu pătratul și rădăcina pătrată sau divizorii acestuia) - adăugarea unei casete de text într-o aplicație simplă, dată, cu o fereastră, o casetă de text și un buton care afișează un rezultat într-o fereastră de mesaje (de exemplu numărul de divizori), astfel încât rezultatul afișat să reflecte ambele valori din casetele de text (de exemplu numărul de divizori comuni)	
CS 3.5. Utilizează elementele de sintaxă și componentele din definiția și apelul subprogramelelor și subprogramelelor predefinite pentru operații uzuale, în implementarea algoritmilor în limbaj de programare	
- folosirea conceptului de transmitere a parametrilor în rezolvarea unei situații practice de calcul al temperaturii maxime, prin apelul unui subprogram Python ce returnează maximul dintre două numere - implementarea sintaxei de definire și apel a unui subprogram Python pentru rezolvarea unei situații practice de calcul al ariei unui teren, folosind ca parametri lungimea și lățimea introduse de utilizator - implementarea subprogramelelor cu funcțiile predefinite abs() și sqrt() în rezolvarea unei situații practice de calcul al distanței dintre două puncte într-un plan, folosind ca parametri coordonatele introduse de utilizator - aplicarea funcțiilor predefinite pentru colecții în rezolvarea unei situații practice de calcul al performanței sportivilor (de exemplu, determinarea mediei, a celei mai mari și celei mai mici valori a scorurilor), prin scrierea unui program Python simplu - aplicarea cunoștințelor despre definirea și apelul subprogramelelor în C++ prin structurarea logică a unei aplicații care calculează media notelor unui elev, prin folosirea unei funcții care primește notele ca parametri și returnează media aritmetică - aplicarea funcției pow() în C++ pentru calcularea sumei finale la economii cu dobândă compusă, folosind formula $A = P \cdot (1 + r/n)^t$, (unde A este suma finală, P este suma inițială, r este rata dobânzii, n este numărul de ori pe an în care dobânda este compusă, t este durata în ani), într-un context real (ex. depunerea a 1000 lei cu dobândă anuală 5% timp de 3 ani)	

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 IX-a	
CS 4.1. Analizează caracteristicile, modul de prelucrare, avantajele și dezavantajele utilizării unor modele conceptuale fundamentale - date simple sau liste, pentru a alege structura adecvată în rezolvarea unor probleme concrete	
- analizarea diferențelor conceptuale dintre stivă, coadă și listă, prin compararea modului de inserare și extragere a elementelor și examinarea avantajelor și limitărilor fiecărui model în contexte reale, precum gestionarea apelurilor telefonice (coadă) versus procesarea comenzilor de tip „anulare ultimă acțiune” (stivă)	

Clasa a IX-a	
- analizarea modului de formare și interpretare a unei liste de frecvențe în studierea obiceiurilor de cumpărare ale clienților, prin descompunerea procesului în etape (colectare, numărare, reprezentare), corelarea valorilor din listă cu comportamentele observate - analizarea modului de funcționare a algoritmilor de parcurgere liniară prin compararea etapelor de verificare, selecție și filtrare a datelor într-o situație practică precum monitorizarea tranzacțiilor zilnice ale unui magazin, pentru a evidenția ordinea logică a operațiilor și dependențele dintre ele - analizarea modului de organizare și prelucrare a unei liste de valori memorate în monitorizarea zilnică a consumului de apă al unei gospodării, prin identificarea relației dintre ordinea datelor, valorile stocate și calculele derivate, precum determinarea consumului mediu și a variațiilor semnificative	
CS 4.2. Analizează comportamentul, aplicabilitatea, avantajele și dezavantajele utilizării unor algoritmi specializați pe clase de probleme pentru prelucrarea numerelor, sortarea sau generarea sistematică a unor secvențe de valori	
- analizarea a doi algoritmi alternativi dați pentru determinarea divizorilor unui număr în vederea estimării numărului de operații efectuate - urmărirea raționamentului rezultat prin operațiile din cadrul unui algoritm pentru a verifica dacă acesta determină obținerea oginditului unui număr natural - analizarea avantajelor utilizării divizorilor unui număr în contexte practic-aplicative (de exemplu, găsirea tuturor posibilităților pentru tăierea unei benzi de L centimetri în bucăți de lungimi egale (numere naturale), fără pierderi de material, posibilitatea de distribuire a N caiete unui număr de elevi astfel încât aceștia să primească același număr de caiete, ambalarea a N produse în cel puțin două pachete astfel încât numărul de pachete să fie minim și în fiecare pachet să fie același număr de produse) - analizarea modului de funcționare a algoritmului Ciurul lui Eratostene în contextul selectării numerelor prime utilizate la codificarea unor culori astfel încât un tablou să poată fi codificat prin produsul culorilor pe care le conține, prin descompunerea etapelor algoritmului, urmărirea modificărilor succesive ale listei de valori și detectarea modului în care procesul de eliminare duce la obținerea numerelor prime finale - analizarea modului de funcționare a algoritmului de exponențiere rapidă în calculul valorilor utilizate la autentificarea tranzacțiilor bancare online, prin descompunerea pașilor algoritmului, urmărirea transformărilor succesive ale valorilor intermediare și detectarea relației dintre reducerea numărului de înmulțiri și creșterea eficienței procesului de calcul - analizarea comportamentului unui algoritm pentru formarea unei liste date (de exemplu, lista primelor n pătrate perfecte consecutive), urmărind raționamentul rezultat prin operațiile din cadrul acestuia pentru a aprecia adecvarea acestuia la cerință - analizarea algoritmului de sortare prin selecția minimului, pentru o lista cu n valori, estimând numărul de comparații pentru un caz favorabil, un caz mediu și un caz nefavorabil - compararea metodei de selecție a minimului cu altă metodă de sortare (de exemplu, metoda bulelor) aplicate pe un set de 200 de punctaje obținute de participanții la olimpiada de informatică din care se vor obține, în ordine descrescătoare, cele mai mari 10 punctaje, din perspectiva numărului de comparații și interschimbări	
CS 4.3. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor	
- analizarea erorilor apărute în programele informatice în scopul încadrării lor într-una dintre categoriile: erori de sintaxă, de logică, de executare - analizarea modulelor unui cod sursă al unui sistem complex de validare a datelor (de ex., un sistem care validează formulare web), punând în evidență contribuția fiecărui modul la procesul general de validare - analizarea avantajelor și dezavantajelor utilizării modulelor în cadrul elaborării unui program - analizarea modului în care interacționează modulele unei aplicații, pe baza unei diagrame care arată componentele acesteia și modul în care interacționează	
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 fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program, în vederea rezolvării eficiente a problemelor informatice	
- analizarea diferențelor de implementare în Python între atribuirea directă a listelor și copierea acestora prin metode specifice (<code>copy()</code>-copiere superficială, <code>slicing</code>-selectarea unei secvențe), în contextul unei aplicații de procesare a datelor financiare, prin observarea efectelor modificării elementelor asupra listelor asociate - analizarea efectelor utilizării metodelor <code>copy()</code> și <code>sort()</code> asupra conținutului listelor, în contextul unei aplicații de evidență a notelor elevilor, prin compararea rezultatelor obținute înainte și după sortare, respectiv copiere - analizarea diferențelor de comportament dintre metodele <code>sort()</code> și <code>reverse()</code> din Python aplicate asupra unei liste de prețuri, în contextul unei aplicații de evidență a vânzărilor, prin observarea și explicarea ordinii obținute și a modificării listei inițiale - analizarea modului în care alegerea tipului de date și a dimensiunii la declararea unui tablou unidimensional C++ influențează gestionarea datelor despre consumul lunar de energie electrică al unui bloc de apartamente, prin examinarea	

Clasa a IX-a	
	corespondenței dintre numărul de elemente, tipul valorilor stocate și spațiul ocupat în memorie, pentru a evidenția importanța unei declarări corecte și eficiente - compararea timpului de executare a unui program care citește datele dintr-un fișier în raport cu același program care citește datele de la tastatură - analizarea avantajelor și dezavantajelor organizării elementelor grafice într-o fereastră, a textelor și a mesajelor asociate, în logica “user-friendly=ușor de utilizat”
CS 4.5. Analizează aplicabilitatea, avantajele și dezavantajele utilizării subprogramelor, în implementarea algoritmilor în limbaj de programare	
	- analizarea descompunerii unui program complex de prelucrare a datelor în subprograme, prin identificarea variabilelor locale și globale și explicarea modului în care acestea influențează funcționarea independentă a fiecărui subprogram - analizarea diferențelor dintre apelul unui subprogram cu parametri și fără parametri, în contextul unui proiect de prelucrare a datelor meteo, prin observarea modului în care se transmit și se utilizează datele - analizarea efectului diferitelor funcții de conversie (int(), float(), str()) asupra acelorași date de intrare, în contextul unei aplicații de procesare a prețurilor produselor folosind subprograme, prin compararea rezultatelor și explicarea diferențelor de tip de date - analizarea modului în care rezultatele funcțiilor min() și max() pot fi influențate de structura și tipul elementelor din colecție, în contextul unei aplicații de prelucrare a prețurilor produselor folosind subprograme, prin compararea rezultatelor pentru liste eterogene - analizarea modului în care parametrii și valorile returnate influențează funcționarea unui subprogram C++ într-o aplicație care determină salariul net al unui angajat, prin compararea apelului prin valoare și apelului prin referință și examinarea avantajelor fiecărei variante

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

Clasa a IX-a	
CS 5.1. Argumentează alegerea unor modele conceptuale fundamentale - date simple sau liste - și a modurilor de prelucrare a acestora pentru rezolvarea unor probleme informatice concrete	
	- evaluarea utilizării adecvate a unei liste, stive sau cozi într-o situație concretă, cum ar fi gestionarea cererilor de suport într-un serviciu de asistență tehnică, prin argumentarea alegerii structurii potrivite în funcție de ordinea de procesare, prioritate și tipul de acces necesar - evaluarea relevanței și acurateței unei liste de frecvențe în prezentarea rezultatelor unei anchete de opinie, prin verificarea modului în care datele reflectă realitatea studiată, argumentarea corectitudinii structurii listei și formularea unei concluzii privind utilitatea sa pentru luarea deciziilor - alegerea unui algoritm de parcurgere liniară fără memorare pentru procesarea datelor provenite dintr-un registru de intrări, prin aprecierea clarității pașilor logici, a modului de acces la date și argumentarea deciziei privind eficiența și corectitudinea abordării alese - evaluarea corectitudinii și relevanței organizării datelor organizate într-o listă utilizată pentru înregistrarea și compararea vânzărilor lunare ale unui produs, prin verificarea completitudinii datelor stocate, a ordinii lor și a modului de interpretare în raport cu scopul analizei comerciale
CS 5.2. Evaluează corectitudinea și eficiența soluțiilor cu algoritmi specializați pe clase de probleme pentru prelucrarea numerelor, sortarea sau generarea sistematică a unor secvențe de valori	
	- evaluarea complexității a doi algoritmi dați pentru determinarea divizorilor unui număr natural stabilind care dintre cei doi algoritmi este mai eficient (de exemplu, parcurgere a posibilor divizori de la 1 până la n, până la $n/2$ sau până la $\sqrt{n}$) - argumentarea faptului că un algoritm ce construiește oglinditul unui număr și apoi oglinditul oglinditului nu conduce totdeauna la obținerea numărului inițial, susținută de teste adecvate și compararea rezultatelor obținute cu cele așteptate - evaluarea eficienței algoritmului Ciurul lui Eratostene în determinarea numerelor prime, în raport cu alt algoritm de generare a numerelor prime până la o valoare dată - evaluarea eficienței algoritmului de exponențiere rapidă în raport cu algoritmul clasic de înmulțiri succesive - justificarea folosirii unui algoritm de generare recurentă a termenilor unei expresii matematice pentru însumarea unor produse, în comparație cu folosirea unui algoritm de calcul direct al fiecărui termen al acesteia (de exemplu pentru a calcula suma $1+1\cdot2+1\cdot2\cdot3+\dots+1\cdot2\cdot3\cdot\dots\cdot n$) - evaluarea eficienței (calculând ordinul de complexitate) unui algoritm de obținere a termenilor șirului lui Fibonacci, pornind de la un termen dat, în ordine descrescătoare, prin două metode: prima metodă memorează ultimul termen afișat apoi îl determină pe cel curent, care îl precede în șir, prin generarea, începând de la 1, în ordine crescătoare, a tuturor termenilor, până la cel căutat, iar a doua metodă generează toți termenii în ordine crescătoare, o singură dată, începând de la 1, până la termenul care îl precede pe cel dat inițial, apoi generarea fiecărui termen curent se face prin diferență, pe baza adaptării regulii de generare specifice

Clasa a IX-a	
- evaluarea corectitudinii unui algoritm care generează o listă cu primii termeni ai șirului lui Fibonacci, pe baza unor teste adecvate care au în vedere și cazuri limită, de exemplu listă vidă, listă cu un singur element, listă cu doar două elemente - argumentarea deciziei de utilizare a metodei de sortare cu listă de frecvențe pentru un sistem de triaj al pacienților din camera de urgență a unui spital, sistem bazat pe ordonarea descrescătoare a gravității stării lor medicale (codificată prin valori de la 1 la 5), ținând cont de caracteristicile datelor (număr mare de elemente, interval numeric limitat) - formularea unei concluzii argumentate privind validitatea și utilitatea metodei bulelor în sortarea zilnică a produselor unui depozit după numărul de zile rămase până la expirare, evidențiind avantajele și limitele metodei în funcție de volumul și variația datelor	
CS 5.3. Evaluează corectitudinea și eficiența algoritmilor care utilizează strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor	
- argumentarea avantajelor descompunerii unei probleme complexe în etape sau subprobleme mai simple (citire date, validare, rezolvare propriu-zisă, afișare rezultat) din perspectiva verificării corectitudinii fiecărui modul - evaluarea timpului de executare a diferiților algoritmi pentru seturi de date de dimensiuni diferite și reprezentarea grafică a rezultatelor - evaluarea comparativă a două soluții ale unei probleme, una modulară și una compactă, argumentând care este mai eficientă - evaluarea corectitudinii și eficienței unei soluții propuse de colegi, oferind feedback privind claritatea și independența modulelor	
CS 5.4. Evaluează corectitudinea și eficiența aplicațiilor care utilizează elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program, în vederea rezolvării eficiente a problemelor informatice	
- justificarea corectitudinii și eficienței utilizării operatorilor de apartenență (in, not in) într-un program de validare a datelor introduse de utilizator, în cadrul unei dezbateri despre criterii de claritate și siguranță în prelucrarea listelor - evaluarea unui program care utilizează obiecte din clasa list, și determină numărul de perechi de valori pare, aflate pe poziții consecutive în listă, pe baza unor teste adecvate care au în vedere comportamentul acestuia pentru elemente care au câte doi vecini, sau se află la fiecare dintre cele două extremități ale listei - evaluarea corectitudinii și pertinentei declarării unei variabile de tip tablou unidimensional utilizate pentru stocarea cantităților de produse dintr-un depozit, prin analizarea tipului de date ales, a dimensiunii tabloului și a modului de denumire a variabilei, precum și argumentarea privind adecvarea acestor alegeri în vederea îndeplinirii scopului aplicației - evaluarea unui program care utilizează tablouri unidimensionale, pe baza unor teste adecvate care au în vedere și cazuri limită, de exemplu acces la o zonă de memorie/poziție în afara celor alocate tabloului - evaluarea corectitudinii unui program cu fișiere text, pe baza unor teste adecvate care au în vedere și cazuri limită, de exemplu fișier inexistent, fișier gol - compararea unei aplicații cu interfață grafică în raport cu aceeași aplicație cu interfață text, evidențiind valențele fiecăreia (avantaje și dezavantaje)	
CS 5.5. Evaluează corectitudinea și eficiența programelor care utilizează subprograme, pentru a rezolva probleme informatice	
- susținerea cu argumente a celor două variante de implementare a aceluiași algoritm (una cu subprograme, alta fără), în cadrul unei dezbateri despre eficiența, claritatea și reutilizarea codului - justificarea alegerii unei variante de definire și apel al unui subprogram Python, din mai multe variante posibile, într-un proiect de calcul al salariului net, prin aplicarea unor criterii de lizibilitate, modularitate și reutilizare a codului - evaluarea acurateței și relevanței utilizării funcțiilor de rotunjire (round(), math.floor(), math.ceil()) într-o aplicație de calcul al totalului facturii, în contextul unei discuții despre criterii de precizie financiară - justificarea utilizării funcției sum() pentru calculul sumei elementelor dintr-o colecție, în comparație cu iterarea explicită, evidențiind avantajele din perspectiva clarității codului și reducerii erorilor, în contextul unei discuții despre bune practici de programare și optimizare a codului - evaluarea corectitudinii unei aplicații cu subprograme, pe baza unor teste adecvate care au în vedere și cazuri limită, de exemplu pentru parametri cu valori inadecvate, lipsa unei valori returnate pentru unele cazuri - evaluarea corectitudinii și clarității sintaxei utilizate în definirea și apelul unui subprogram care calculează reducerea aplicată unui produs în funcție de procentul introdus de utilizator, prin verificarea respectării convențiilor de numire, a tipului de retur și a utilizării parametrilor, și argumentarea privind calitatea soluției	

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

Clasa a IX-a	
CS 6.1. Proiectează algoritmi personalizați care utilizează modele conceptuale fundamentale - date simple sau liste pentru organizarea și prelucrarea eficientă a datelor, utilizând algoritmi de bază, în vederea rezolvării unor probleme	
- proiectarea unui model conceptual care combină principiile listelor, stivelor și cozilor pentru optimizarea fluxului de sarcini într-un centru de livrări, printr-un sistem de organizare a comenzilor care stochează, prioritizează și eliberează elementele conform regulilor de acces definite pentru fiecare tip de structură - proiectarea unui model conceptual de listă de frecvențe pentru monitorizarea numărului de erori raportate zilnic, pentru fiecare tip posibil, într-un centru de asistență tehnică, prin stabilirea structurii listei, a modului de înregistrare a fiecărei erori și a regulilor de actualizare a frecvențelor, astfel încât modelul să poată fi ulterior implementat în limbaj de programare - proiectarea unui algoritm de parcurgere liniară pentru determinarea și prelucrarea automată a valorilor neconforme dintr-o listă de date financiare, prin stabilirea pașilor de acces succesiv la elemente, definirea condițiilor de verificare și formularea logicii generale de procesare, care va constitui ulterior baza implementării în limbaj de programare - proiectarea unei liste de valori memorate pentru urmărirea evoluției zilnice a nivelului de poluare într-un oraș, prin definirea modului de colectare și stocare a valorilor unui anumit tip de noxe, stabilirea algoritmilor de parcurgere pentru calculul valorii medii și identificarea zilelor critice cu valori care depășesc media, ca bază pentru o viitoare implementare în limbaj de programare	
CS 6.2. Proiectează soluții cu aplicarea personalizată a algoritmilor specializați pe clase de probleme pentru prelucrarea numerelor, sortarea sau generarea sistematică a unor secvențe de valori	
- proiectarea unui program ce implementează operații cu câte două fracții date prin numărătorul și numitorul lor (simplificare, adunare, scădere, înmulțire, împărțire), utilizând personalizat algoritmi de divizibilitate - rezolvarea unei probleme ce determină cel mai mic multiplu comun a două numere pe baza celui mai mare divizor comun al lor, pentru obținerea duratei minime de timp după care două semafoare trec simultan la aceeași culoare, cunoscându-se, pentru fiecare, timpul de menținere a fiecărei culori (roșu și verde) - conceperea unui algoritm care determină numărul de persoane care sunt născute într-o anumită lună a anului pe baza unei liste de coduri numerice personale date - dezvoltarea unui program care aplică algoritmul Ciurului lui Eratostene pentru selectarea numerelor prime până la n, utilizate la generarea identificatorilor unici ai participanților la „Maratonul numerelor prime”, prin proiectarea logicii algoritmului folosind liste, testarea pentru diferite valori ale lui n și prezentarea modului în care soluția asigură corectitudinea identificatorilor - crearea unui algoritm eficient care afișează primii n termeni ai unui șir recurent, prin determinarea formulei termenului general, cunoscându-se primii termeni și relația de recurență (de exemplu, pentru $a_0=3$ și relația de recurență $a_n=2 \cdot a_{n-1}$, pentru $n>0$, se obține $a_n=3 \cdot 2^n$) - implementarea unui program pentru gestionarea solicitărilor de mentenanță pentru o companie de lifturi, care își sortează cererile în funcție de tipul defecțiunii (codificat cu valori de la 1 la 8), obținându-se în final lista cererilor ordonate - dezvoltarea unui program care utilizează metoda bulelor pentru organizarea programărilor zilnice ale unei clinici medicale în ordinea crescătoare a duratei consultațiilor, prin proiectarea algoritmului, testarea funcționării sale pe un set de date simulate și realizarea unei afișări care să evidențieze clar modul de aplicare al metodei	
CS 6.3. Proiectează modular algoritmi personalizați pentru rezolvarea problemelor	
- descompunerea unei aplicații în module, în cadrul unui grup, în care fiecare modul este abordat de câte un membru al grupului, și integrarea acestor rezolvări, ulterior, într-un produs unitar - conceperea unei diagrame generale pentru rezolvarea modulară a unei probleme, evidențiind componentele și conexiunile dintre ele - completarea unei aplicații existente (de exemplu: o aplicație de gestionare a bibliotecii școlare, un sistem de evidență a prezenței, un registru digital etc.), a cărei descriere este dată, cu un modul nou, care adaugă o funcționalitate suplimentară (de exemplu, raportare, notificări, export de date, interfață de administrare etc.), respectă principiile de proiectare modulară și poate fi integrat în structura existentă	
CS 6.4. Implementează aplicații care integrează în mod personalizat elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program, în vederea realizării unor soluții funcționale și performante	
- elaborarea unei aplicații Python de gestiune a listelor de produse dintr-un magazin online, prin integrarea operatorilor de acces, apartenență, concatenare și relaționare pentru actualizarea și compararea automată a stocurilor din mai multe surse - proiectarea unei aplicații Python pentru administrarea unei liste de produse dintr-un magazin, utilizând metodele clasei list (<code>append()</code>, <code>insert()</code>, <code>pop()</code>, <code>count()</code>, <code>index()</code>, <code>copy()</code>, <code>sort()</code>) pentru a adăuga, elimina, sorta și analiza elementele din stoc - elaborarea unei aplicații Python pentru gestionarea unei liste dinamice de sarcini (to-do list), prin identificarea și utilizarea metodelor adecvate (<code>append()</code>, <code>remove()</code>, <code>sort()</code>, <code>reverse()</code>, <code>copy()</code>, <code>clear()</code>) pentru adăugarea, eliminarea, ordonarea și resetarea elementelor	

Clasa a IX-a
- crearea unui tablou unidimensional care stochează valorile temperaturilor zilnice dintr-o săptămână și utilizarea operatorului [] pentru accesarea și modificarea elementelor, astfel încât să poată fi actualizată temperatura dintr-o zi selectată de utilizator - proiectarea unei aplicații de exploatare cu valențe practice a unui fișier text (de generare a unor serii aleatoare de numere folosind biblioteca random urmată de numărări și rezultate statistice pentru o astfel de serie generată) - implementarea unei aplicații Python pentru afișarea, într-o fereastră, cu fundal colorat, a câte unui termen aparținând unui șir generat pe baza unei relații de recurență, utilizând două obiecte din clasa Button, cu proprietatea text completată sugestiv, pentru a marca pornirea, respectiv oprirea procesului de obținere a termenilor șirului
CS 6.5. Implementează aplicații în limbaj de programare care integrează subprograme personalizate, pentru a modulariza și organiza eficient codul în cadrul soluțiilor informatice
- construirea unui mini-proiect de tip aplicație modulară Python (de exemplu, gestionarea unui catalog școlar), prin proiectarea și implementarea propriilor subprograme care utilizează parametri și returnează valori, cu prezentarea funcționalității în fața clasei - proiectarea unei aplicații modulare Python de tip calculator personalizat (ex. conversii valutare, conversii de unități), prin definirea și apelarea mai multor subprograme proprii, utilizând parametri și valori returnate relevante pentru funcționalitatea dorită - construirea unui mini-proiect de tip aplicație modulară Python pentru conversia și prelucrarea valorilor numerice (de exemplu, o aplicație care convertește temperaturi, distanțe sau sume de bani), prin integrarea mai multor funcții predefinite de calcul și conversie (abs(), round(), int(), float(), sqrt()) - proiectarea unei aplicații modulare Python care gestionează o colecție de date reale (de exemplu, temperaturi zilnice, vânzări lunare, note școlare), utilizând funcțiile len(), min(), max(), sum() pentru generarea unui raport sintetic cu informații despre datele analizate - conceperea unui subprogram în C++ care determină scorul final al unui jucător într-un joc video pe baza mai multor criterii (puncte, timp, bonusuri), prin definirea completă a funcției cu parametri și valoare de retur, apelarea sa în programul principal și formularea logicii de implementare

CONȚINUTURI ALE ÎNVĂȚĂRII**Clasa a IX-a**

Domenii de conținut	Conținuturi
1. Organizarea conceptuală a datelor	1.1. Modelul conceptual liniar – listă - caracteristici, din punctul de vedere conceptual, ale unei liste și ale cazurilor particulare date de tipul de acces (stivă, coadă, respectiv cu acces direct, cu acces secvențial), rolul avut în prelucrarea datelor (listă de frecvențe); - repere pentru parcurgerea elementelor și pentru aplicarea algoritmilor de bază pentru prelucrarea datelor organizate liniar, cu sau fără memorare.
2. Strategii de rezolvare a problemelor	2.1. Principii de elaborare a unui program - caracteristici ale gândirii computaționale și ale principalelor etape ale elaborării unui program (analiză, proiectare, implementare, testare și depanare), repere pentru aplicarea acestor etape în rezolvarea problemelor; - caracteristici de bază, avantaje și limite ale principalelor moduri de reprezentare a algoritmilor: blocuri grafice, pseudocod, limbaj de programare (cu interpretor sau compilator); - repere pentru proiectarea modulară a algoritmilor; - criterii de elaborare a testelor (pentru validarea datelor și pentru verificarea comportamentului algoritmului), repere pentru urmărirea evoluției valorilor variabilelor în scopul identificării și tratării erorilor; - eficiența algoritmilor: caracteristici ale eficienței din punctul de vedere al spațiului de memorie și din punctul de vedere al timpului de executare, criterii pentru determinarea ordinului de complexitate a unui algoritm polinomial (notația O); - caracteristici ale unor modalități de bază de comunicare cu programul: interfață de tip consolă, interfață grafică (ferestre, butoane, etichete, casete text), fișiere. 2.2. Prelucrări ale datelor numerice - operații specifice cu cifrele unui număr (acces la o cifră a unui număr, adăugare a unei cifre la stânga unui număr, adăugare a unei cifre la dreapta unui număr), determinarea unui divizor/multiplu al unui număr; - repere pentru prelucrarea numerelor (de exemplu parcurgerea cifrelor unui număr, a divizorilor unui număr, descompunere a unui număr în factori primi) și aplicarea algoritmilor de bază; - algoritmi elementari pentru determinarea celui mai mare divizor comun (algoritmul lui Euclid cu scăderi repetate, respectiv cu împărțiri repetate); - repere pentru transformarea unui număr dintr-o bază de numerație în altă bază de numerație; - ciurul lui Eratostene - algoritm eficient pentru determinarea unor numere prime: caracteristici și repere pentru reprezentarea algoritmului utilizând liste; - algoritm de exponențiere rapidă: caracteristici și repere pentru reprezentarea algoritmului. 2.3. Metode de generare sistematică a elementelor unei liste - repere pentru generarea unor secvențe de valori (de exemplu secvențe cu proprietăți date, termeni ai unor expresii matematice, termeni ai unor șiruri recurente) și aplicarea algoritmilor de bază. 2.4. Metode de sortare a elementelor unei liste - caracteristici ale metodei de sortare prin selecția minimului și repere pentru aplicarea metodei; - caracteristici ale metodei de sortare cu listă de frecvențe și repere pentru aplicarea metodei; - caracteristici ale metodei bulelor și repere pentru aplicarea metodei.

Domenii de conținut	Conținuturi
3. Memorarea datelor și organizarea codului în limbaj de programare	3.1. Subprograme - caracteristici, rol; - concepte de bază și caracteristici: antet, corp, variabile locale, variabile globale, transmitere prin intermediul parametrilor, returnare a rezultatelor, apel, mecanism de executare; - sintaxă pentru definiția și apelul unui subprogram în Python; - caracteristici și repere de utilizare a unor subprograme predefinite în Python pentru operații matematice uzuale (de exemplu radical, valoare absolută, parte întreagă, rotunjire) și conversii; - caracteristici și repere de utilizare a unor subprograme predefinite în Python pentru colecții de valori (determinarea numărului de elemente, a minimului, maximumului, sumei unor valori); - sintaxă pentru definiția și apelul unui subprogram în C++; - caracteristici și repere de utilizare a unor subprograme predefinite din C++ pentru operații matematice uzuale (radical, valoare absolută, parte întreagă, rotunjire, ridicare la putere); - caracteristici și repere de utilizare a unor subprograme predefinite din C++ pentru colecții de valori (determinarea a minimului, maximumului, sortare). 3.2. Introducere în programarea orientată pe obiecte în limbaj de programare - noțiuni de bază necesare utilizării unor clase predefinite: clasă, membri ai clasei (date și metode), obiecte, biblioteci; - instanțiere a unei clase predefinite, acces la membrii unui obiect. 3.3. Fișiere text - caracteristici, principii de lucru (deschidere, închidere, transfer de date); - funcție pentru deschiderea unui fișier în Python, clasa predefinită TextIOWrapper din Python pentru prelucrarea fișierelor: caracteristici, metode de bază (pentru citire, scriere, închidere); - clase și metode uzuale din C++ pentru prelucrarea unui fișier: deschidere, citire, scriere, închidere; - repere pentru identificarea și utilizarea altor clase și metode, adecvate pentru prelucrarea fișierelor și pregătirea datelor citite din fișiere. 3.4. Biblioteca tkinter din Python pentru interfețe grafice - clase, caracteristici, funcții și metode de bază pentru interfețe grafice simple cu ferestre, butoane, etichete, casete text (Tk, Label, Button, Entry, Text, Frame, Canvas), afișare a mesajelor, comportament (pack, grid, place, get); - repere pentru identificarea și utilizarea altor clase și metode pentru realizarea interfețelor grafice. 3.5. Clasa list din Python – clasă predefinită pentru memorarea unei liste - caracteristici; - operatori specifici pentru acces la un element, apartenență, non-apartenență, concatenare, multiplicare, relaționare; - metode ale clasei pentru operații de bază: determinare a primei poziții a unei valori, numărare a aparițiilor unei valori, ștergere a elementului de la o anumită poziție, inserare a unei valori într-o anumită poziție, adăugare a unui element, copiere, sortare; - repere pentru identificarea și utilizarea altor metode ale clasei, adecvate pentru prelucrarea listelor. 3.6. Structuri de date în C++ pentru memorarea listelor – tablouri unidimensionale - caracteristici, declarare a variabilelor de tip tablou unidimensional, operator specific pentru acces la un element.

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 măsoară? 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.

Un context de învățare poate viza simultan mai multe competențe, mai ales atunci când acestea sunt interconectate. De exemplu, pentru rezolvarea unei probleme cu ajutorul calculatorului se vizează, în cadrul aceleiași activități de învățare, atât competențe de nivel cognitiv scăzut (identificarea principalelor caracteristici ale modelelor conceptuale și operaționale implicate, explicarea principiilor care stau la baza acestora, în contextul particular al problemei, utilizarea lor în rezolvare) dar și competențe de nivel cognitiv ridicat (analizarea caracteristicilor modelelor conceptuale și operaționale specifice problemei, evaluarea corectitudinii și eficienței soluțiilor propuse și elaborarea algoritmului personalizat).

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, algoritimizarea, 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.

Conform sugestiilor metodologice din programa școlară de *informatică și TIC* pentru gimnaziu, la clasele a VII-a și a VIII-a pentru formarea competențelor specifice s-a putut utiliza un limbaj de programare dintr-o listă care cuprinde, de exemplu, propuneri ca Python, C, C++. Deoarece alegerea limbajului de programare a fost la latitudinea profesorului, este necesară **o armonizare a acestor limbaje de programare la trecerea în clasa a IX-a, pentru ca toți elevii să utilizeze limbajul de programare precizat în programa școlară în vigoare pentru această clasă**. Astfel, se recomandă prezentarea, la început, a unor elemente de corespondență a principalelor instrumente de reprezentare a algoritmilor în pseudocod, blocuri grafice, limbaje de programare Python și C++: citire și afișare a datelor, biblioteci, comenzi și scripturi, instrucțiuni, variabile și tipuri de date de bază, operatori de bază, medii integrate pentru dezvoltarea programelor (IDE) și funcționalități ale acestora.

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).

Pentru specializarea matematică-informatică, la clasele cu predarea disciplinei informatică în regim intensiv, **limbajele de programare pentru implementare sunt Python și C++**, de exemplu în una dintre variantele:

- în paralel (alocând pentru practică anumite ore, săptămânal, fiecărui limbaj, ca în cazul limbilor străine, sau exemplificând alternativ implementări în cele două limbaje), evidențiind pe parcurs asemănările și deosebirile dintre acestea;
- succesiv, realizându-se o recapitulare a unor concepte, în celălalt limbaj, conform programei școlare.

Predarea elementelor de limbaj specifice Python și C++ în paralel oferă o abordare solidă, pe baza unui plan didactic eficient, având în vedere consolidarea conceptelor comune și susținerea învățării prin comparație (observând asemănări de logică și structură, diferențe de sintaxă, o înțelegere mai profundă a conceptelor de bază, cum ar fi gestionarea automată sau manuală a memoriei), evidențierea complementarității conceptuale (Python oferă o sintaxă simplă, ușor de citit, care permite elevilor să se concentreze pe concepte fără să fie pus accent pe detalii tehnice, iar C++ facilitează înțelegerea profundă a mecanismelor din spatele limbajului — tipuri de date, gestionarea memoriei, pointeri, compilare), formarea unei gândiri flexibile în programare, evidențiindu-se faptul că limbajul este doar un instrument, iar gândirea logică este universală, facilitându-se astfel adaptabilitatea la orice alt limbaj viitor.

Un exemplu, orientativ, de ordine de abordare a conținuturilor pentru clasa a IX-a, specializarea matematică-informatică, pentru clase cu predarea disciplinei informatică în regim intensiv, cu o abordare în paralel a celor două limbaje de programare:

1. *Strategii de rezolvare a problemelor. Principii de elaborare a unui program*
2. *Strategii de rezolvare a problemelor. Prelucrări ale datelor numerice*
3. *Memorarea datelor și organizarea codului în limbaj de programare. Subprograme (în Python și C++)*
4. *Memorarea datelor și organizarea codului în limbaj de programare. Introducere în programarea orientată pe obiecte în limbaj de programare*
5. *Memorarea datelor și organizarea codului în limbaj de programare. Biblioteca Tkinter din Python pentru interfețe grafice*
6. *Memorarea datelor și organizarea codului în limbaj de programare. Fișiere text (în Python și C++)*
7. *Organizarea conceptuală a datelor. Modelul conceptual liniar – listă*
8. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa list din Python și tablouri unidimensionale în C++*
9. *Strategii de rezolvare a problemelor. Metode de generare sistematică a elementelor unei liste*
10. *Strategii de rezolvare a problemelor. Metode de sortare a elementelor unei liste*

Se recomandă ca în cadrul temei *Strategii de rezolvare a problemelor. Principii de elaborare a unui program* să se prezinte **doar succint** unele moduri de reprezentare a algoritmilor (blocuri grafice, schemă logică), punând în evidență utilizarea lor pe scară largă în diferite domenii, urmând ca ulterior să fie pus accentul pe pseudocod și limbajele de programare precizate în programă (în funcție de nivel și specializare), în etapa de proiectare, respectiv de implementare. Astfel, 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 ca schemă logică, 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.

În ceea ce privește prelucrarea numerelor se recomandă utilizarea exemplelor numerice și a exercițiilor practice pentru explorarea operațiilor cu cifrele unui număr, evidențiind logica accesului la acestea și compunerii numerelor. Identificarea divizorilor, descompunerea în factori primi se va realiza cu accent pe înțelegerea logicii pas cu pas, nu pe calcule complicate. Algoritmii pentru determinarea celui mai mare divizor comun vor fi prezentați intuitiv, de exemplu prin observarea împărțirilor succesive, fără formalizare matematică excesivă.

Se recomandă introducerea noțiunii de subprogram pornind de la analogii din viața reală („o rețetă”, „o instrucțiune care poate fi refolosită”) pentru a înțelege ideea de modularitate.

Elevii pot experimenta definirea și apelul de funcții prin exemple practice (calculul unei medii, determinarea unei valori maxime), iar funcțiile predefinite vor fi prezentate ca instrumente utile pentru scurtarea și claritatea codului.

Programarea orientată pe obiecte se poate introduce la clasa a IX-a prin analogii intuitive cu lumea reală, evidențiind faptul că orice obiect are proprietăți (date) și comportamente (metode), pentru a facilita înțelegerea conceptelor de clasă și membri ai clasei. Predarea are în vedere doar noțiuni **strict necesare pentru utilizarea, la nivel elementar, a unor clase predefinite**.

Profesorul poate exemplifica trecerea de la interfețele de tip consolă la cele grafice prin construirea unor mini-aplicații interactive. Crearea interfețelor grafice poate fi introdusă prin proiecte simple și atractive (de exemplu: calculator, aplicație de notițe, joc de tip „quiz”). Activitățile de învățare pot urmări legătura dintre interacțiunea elevului cu programul și comportamentul acestuia, stimulând interesul pentru proiectare și creativitate.

Lucrul cu fișiere text se poate aborda prin activități practice de învățare care ilustrează procesul complet de deschidere, citire, scriere și închidere a unui fișier, folosind exemple apropiate de experiența elevilor (liste de elevi, jurnale, notițe).

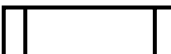
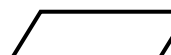
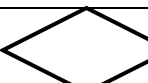
În baza modelului conceptual liniar de organizare a datelor, listele sunt privite ca **o succesiune de date, care vor fi sau nu memorate**; acestea nu sunt abordate ca liste înălțuite la clasa a IX-a.

Modelul conceptual de organizare a datelor de tip listă poate fi introdus pornind de la contexte familiare (de exemplu: liste de cumpărături, clasamente, cozi de așteptare), pentru a facilita înțelegerea ideii de organizare și acces la date. Stiva și coada pot fi explicate intuitiv prin activități practice (de exemplu aranjarea unor obiecte fizice) pentru evidențierea regulilor de acces la elemente, adăugare și eliminare a acestora. Parcurgerea și prelucrarea elementelor se pot exercita prin activități concrete: numărarea elementelor, identificarea pozițiilor, ordonarea după criterii simple; algoritmi de bază pot fi aplicați inițial pe liste cu număr mic de elemente, pentru înțelegerea logicii, nu pentru complexitate.

Elevii pot explora metodele și proprietățile clasei list din Python prin activități interactive care simulează manipularea unor colecții de obiecte (adăugare, ștergere, sortare). Se recomandă utilizarea metodelor uzuale (de exemplu append, remove, sort) în contexte concrete, punând accent pe experimentare și descoperire a efectului comenzii. Exemplele vor fi scurte, intuitive, centrate pe calcule matematice simple.

Pentru algoritmi de sortare prin selecția minimului, listă de frecvențe și metoda bulelor se recomandă utilizarea animațiilor digitale sau simulărilor cu scopul de a vizualiza modul de funcționare pas cu pas al algoritmilor, însoțite de discuții despre situații practice unde este necesară ordonarea.

Pentru reprezentarea schemelor logice, simbolurile uzuale sunt:

Simbol	
	Linie de flux: reprezintă trecerea controlului între formele conectate
	Bloc de proces: reprezintă un bloc de calcul
	Bloc de subprogram/modul: reprezintă un apel de subprogram/modul care este reprezentat printr-o schemă logică separată
	Bloc de intrare/ieșire: reprezintă intrarea (CITIRE) sau ieșirea (SCRIERE) datelor
	Bloc de decizie: reprezintă o decizie care are ca rezultat două ieșiri, reprezentând cele două valori posibile (Da/Nu sau Adevărat/Fals)
	Bloc terminal: reprezintă momentul de început (START) sau de final (STOP) al procesului
	Bloc conector: reprezintă conectarea a două sau mai multe puncte ale schemei logice

Pentru reprezentarea algoritmilor în pseudocod, convențiile uzuale sunt:

Indentare: liniile sunt indentate pentru a indica faptul că sunt conținute într-o structură dintr-o linie anterioară

Operațiile de intrare și ieșire sunt precizate prin comenzile **citește <identificator sau identificatori, separați prin virgulă>** respectiv **scrie <expresie sau expresii, separate prin virgulă>**

Operațiile de atribuire sunt precizate în următorul format: **<identificator> ← <expresie>**

Structurile de control sunt precizate în următoarele formate:

- structură de decizie

```

dacă <condiție> atunci
  <Instrucțiuni>
[altfel
  <Instrucțiuni>]

```

sau

```

dacă <condiție> atunci
  <Instrucțiuni>
[altfel
  <Instrucțiuni>]
sfârșit dacă

```

- structură repetitivă cu test inițial

```

cât timp <condiție> execută
  <Instrucțiuni>

```

sau

```

cât timp <condiție> execută
  <Instrucțiuni>
sfârșit cât timp

```

- structură repetitivă cu test final

```

repetă
  <Instrucțiuni>
până când <condiție>

```

- structură repetitivă cu număr cunoscut de pași (unde valorile contorului includ limitele: <expIni>, respectiv <expFin>)

```

pentru <contor><<expInit>,<expFin>[,<pas>] execută
  <Instrucțiuni>

```

sau

```

pentru <contor><<expInit>,<expFin>[,<pas>] execută
  <Instrucțiuni>
sfârșit pentru

```

Subprogramele/modulele sunt definite în formatele:

```

subprogram <identificator> [(<parametri formali>)] [returnează <rezultate sau tipuri de date>]
  <Instrucțiuni>

```

sau

```

subprogram <identificator> [(<parametri formali>)] [returnează <rezultate sau tipuri de date>]
  <Instrucțiuni>
sfârșit subprogram
Cu apelul
Apel <identificator> [(<listă de parametri efectivi>)]

```

Un **element al unei liste indexate** se accesează precizând poziția ca indice (sau indici, separați prin virgulă), ai variabilei de tip listă: de exemplu a_i sau $a_{i,j}$.

Comentariile sunt precedate de două bare oblice // înainte și continuă până la sfârșitul liniei

Pentru operații aritmetice se utilizează simboluri standard ale operatorilor aritmetici: + (adunare), – (scădere), * (înmulțire), / (împărțire), ^ (ridicare la putere), $\sqrt{}$ (rădăcină pătrată), pentru partea întreagă a unui număr real se utilizează încadrarea expresiei între paranteze drepte, iar pentru restul împărțirii a două numere întregi se utilizează operatorul %.

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://repl.it.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 Național Militar „Alexandru Ioan Cuza”, Constanța, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forțelor 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 Național Militar „Dimitrie Cantemir”, Breaza, județul Prahova
TEGLAȘ Maria	profesor, gradul didactic II	Colegiul Național 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

Anexa nr. 9

Programa școlară
pentru disciplina

Informatică

Clasa a X-a

Curriculum de specialitate (CS)

*Filiera teoretică, profilul real, specializarea matematică-informatică,
clase cu predarea disciplinei informatică în regim intensiv*

Învățământ liceal

- 2026 –

Anexa nr. 9**Programa școlară pentru disciplina *Informatică*, clasa a X-a, curriculum de specialitate (CS)**

- *Filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei informatică în regim intensiv*

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 matematică - informatică, clase cu predarea disciplinei *informatică* în regim intensiv, conform O.M.E.C. nr. 6873/2025 alocarea orară este:

- clasa a IX-a – 4 ore/săptămână, dintre care două ore pentru studiu teoretic și două ore pentru activități practice;
- clasa a X-a – 4 ore/săptămână, dintre care două ore pentru studiu teoretic și două ore pentru activități practice;
- clasa a XI-a – 7 ore/săptămână, dintre care patru ore pentru studiu teoretic și trei ore pentru activități practice;
- clasa a XII-a – 7 ore/săptămână, dintre care patru ore pentru studiu teoretic și trei ore pentru activități practice.

Activitățile practice sunt desfășurate obligatoriu în laboratorul de informatică, iar în baza Ordinului ministrului educației și cercetării nr. 6873/2025, pentru filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei *informatică* în regim intensiv, activitățile practice se desfășoară cu colectivul de elevi organizat pe grupe.

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 (Ordinul 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.

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 limbajelor de programare Python și 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 de programare Python 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 asigură o curbă de învățare blândă și 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. Limbajul 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 - neliniare, liniare, asociative, mixte, pentru a structura și accesa date în vederea prelucrării acestora	
- definirea conceptului de mulțime în contextul determinării drepturilor distincte de acces care sunt atribuite unui grup de utilizatori cu roluri diferite, unde fiecare drept apare o singură dată, chiar dacă e atribuit mai multor persoane, iar ordinea înregistrării acestora nu contează - enumerarea operațiilor de bază asupra mulțimilor (reuniune, intersecție, diferență, egalitate, incluziune, testare apartenență), în contextul în care se cunoaște lista obiectelor transportate cu două rucsacuri și se cere să fie identificate obiectele comune, cele conținute de un singur rucsac etc. - precizarea caracteristicilor unei mulțimi în contextul gestionării utilizatorilor înregistrați pe platforme diferite pentru a evita dublurile - 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 - 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 - definirea conceptului de structură asociativă (dicționar) cu evidențierea faptului că se rețin perechi cheie-valoare și se permite accesul rapid la date prin intermediul cheilor, de exemplu, într-o aplicație de traducere, în care fiecare cuvânt este o „cheie”, iar traducerea sa este „valoarea” sau în contextul asocierii dicționarului cu agenda telefonică în care fiecărui număr de telefon (cheie) îi corespunde un nume (valoare) - enumerarea cheilor și a valorilor asociate dintr-un dicționar pentru un formular online, în care fiecare câmp completat (de exemplu, „nume”, „vârstă”, „email”) este asociat cu valorile corespunzătoare - 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; - identificarea caracteristicilor unui model mixt, de tipul dicționar de liste și a modului de organizare a datelor în contextul evidenței școlare a claselor, în care fiecare clasă are o listă de elevi - enumerarea metodelor de acces și parcurgere a datelor într-o listă de dicționare în contextul gestiunii unor comenzi, fiecare comandă fiind reprezentată printr-un dicționar (prin asocierea produs-cantitate) pentru a genera facturile corespunzătoare - enumerarea caracteristicilor tuplurilor în contextul utilizării lor pentru stocarea coordonatelor geografice (latitudine, longitudine) într-un sistem de navigație care necesită date ce nu trebuie să fie modificate accidental - identificarea caracteristicii tuplurilor de reprezentare a unor perechi fixe de valori în contextul memorării culorilor ca triplete RGB, într-o aplicație de grafică - enumerarea caracteristicilor modelelor conceptuale liniare tuplu și listă, în contextul reprezentării unui punct fix de pe o hartă (de exemplu, un oraș, ale cărui coordonate nu pot fi modificate) și a unui punct mobil de pe hartă (care se actualizează, de exemplu, poziția unui robot care se mișcă)	
CS 1.2. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea listelor, a șirurilor de caractere	
- recunoașterea caracteristicilor metodei căutării binare cu specificarea condiției necesare, în contextul localizării rapide a unui contact într-o agendă telefonică ordonată alfabetic - enumerarea etapelor metodei căutării binare în contextul localizării unui titlu într-o listă ordonată de cărți dintr-o bibliotecă digitală - 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 - 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 - enumerarea unor metode simple de criptare (de exemplu, cifrul lui Cezar, substituția monoalfabetică) în contextul utilizării unor tehnici de codificare pentru a ascunde indicii într-un joc de tip „vânătoare de comori”	

Clasa a X-a	
- identificarea caracteristicilor specifice fiecărei metode de criptare (de exemplu, cifrul lui Cezar - deplasarea literelor, substituție monoalfabetică - înlocuire literă cu literă) în contextul analizării unui mesaj codificat, observând dacă literele sunt deplasate, înlocuite sau reordonate - recunoașterea etapelor algoritmilor de umplere (Flood Fill), în contextul determinării punctelor în care poate ajunge o persoană care traversează un teren cu obstacole, reprezentat prin linii și coloane de valori 0 și 1, unde 1 indică un obstacol și 0 o zonă accesibilă	
CS 1.3. Identifică specificul, caracteristicile și etapele unor strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor	
- definirea caracteristicilor metodei Divide et impera în contextul unei discuții ghidate în care sunt reliefate avantajele metodei prin analogie cu fragmentarea unei activități complexe în subactivități realizate de echipe diferite, urmate de reunirea rezultatelor - enumerarea pașilor specifici metodei Divide et impera (divizarea problemei, rezolvarea subproblemelor, combinarea soluțiilor) în contextul aplicării acestei strategii la algoritmul de determinare a valorii maxime dintr-un șir de numere memorate - 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 - 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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice	
- recunoașterea sintaxei corecte de inițializare a unei mulțimi (set) în contextul în care sunt adunate obiecte diferite (de exemplu, tipuri de fructe) într-o colecție în care elementele sunt unice, cu evidențierea modului în care utilizarea acoladelor și a elementelor separate prin virgule permite crearea unei mulțimi fără dubluri și fără ordine prestabilită - identificarea operatorului pentru verificarea apartenenței unui element la o mulțime în Python în contextul scrierii unei secvențe de instrucțiuni prin care se verifică dacă o persoană este deja înscrisă pe lista participanților la un eveniment, fără a fi necesară parcurgerea întregii liste - 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.) - 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 - 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 - 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 - recunoașterea notării specifice cu acolade și a separatorului între cheie și valoare în cazul dicționarelor, prin analogie cu organizarea informațiilor într-o listă de contacte: numele persoanei urmat de simbolul „:”, apoi de numărul de telefon, elementele fiind grupate între acolade - enumerarea operațiilor de bază asupra dicționarelor (creare, adăugare, ștergere, actualizare) prin analogie cu actualizarea listei de contacte din agenda telefonică: crearea unei liste de contacte, adăugarea unui contact nou, ștergerea unui contact existent, modificarea numărului de telefon al unei persoane - 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 - 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 - identificarea modului de inițializare a unui tuplu în Python, pentru un punct fix pe o hartă, dat prin coordonatele (x,y) - recunoașterea sintaxei de accesare a componentelor unui tuplu în Python, în contextul accesării abscisei sau ordonatei unui punct fix de pe o hartă, reprezentat printr-un tuplu	

Clasa a X-a	
- enumerarea unor metode uzuale ale clasei string din C++ (precum insert, erase, find) în contextul prelucrării textului unui mesaj care trebuie modificat înainte de a fi trimis într-o conversație online - precizarea elementelor de sintaxă pentru definirea unei structuri în C++ în contextul abstractizării unor obiecte din viața reală (cum ar fi un telefon cu specificațiile sale sau fișa cu date personale ale unui elev) ale căror caracteristici pot fi memorate cu ajutorul structurilor, pe baza faptului că fiecare obiect are mai multe caracteristici, de diferite tipuri - identificarea operatorilor utilizați pentru accesarea unui câmp dintr-o structură în C++, în contextul prelucrării datelor unei structuri care reprezintă un obiect real, precum un telefon, un automobil sau un produs dintr-un catalog	
CS 1.5. Identifică elementele de sintaxă și componentele din definiția și apelul subprogramelor recursive, pentru a le utiliza în implementarea algoritmilor în limbaj de programare	
- enumerarea caracteristicilor principale ale unui subprogram recursiv, în contextul elaborării unui astfel de subprogram pentru calculul lui $n!$ definit prin relația recurentă cunoscută - recunoașterea termenilor „recursivitate directă” și „recursivitate indirectă” prin asocierea lor cu definițiile corespunzătoare - recunoașterea sintaxei corecte pentru definirea unui subprogram recursiv în Python, în contextul calculării sumei cifrelor unui număr - enumerarea condițiilor pe care trebuie să le îndeplinească un subprogram recursiv pentru a funcționa corect, menționând necesitatea unui caz de oprire, a unei transformări care reduce progresiv problema și a unui apel recursiv care aplică aceeași procedură până la atingerea condiției de oprire, în contextul calculării factorialului unui număr - descrierea elementelor specifice sintaxei pentru declararea unei funcții recursive în C++ - recunoașterea sintaxei pentru definirea unei funcții recursive în C++ în contextul adunării primelor n numere naturale, prin alegerea unei variante de sintaxă din două posibile	

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 - neliniare, liniare, asociative, mixte, pentru a le gestiona eficient	
- explicarea caracteristicilor mulțimilor și listelor, în contextul organizării și accesării elementelor acestora în situații cotidiene, de exemplu, o listă de persoane care așteaptă la rând și determinarea clienților distincți ai unei companii de telefonie, pe baza unei mulțimi de abonamente (în condițiile în care un client poate avea mai multe astfel de abonamente) - explicarea modului în care mulțimile sunt utilizate pentru clasificarea pe categorii a produselor într-un magazin online, evidențiind cum apartenența la o categorie permite organizarea și filtrarea eficientă a articolelor - 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 - 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 - explicarea rolului cheilor unice și a valorilor asociate în dicționare, în contextul serviciului de evidența populației, unde nu pot exista două CNP-uri identice (chei unice), iar modificarea numelui unei persoane actualizează valoarea corespunzătoare cheii respective - explicarea modului în care un dicționar poate fi folosit pentru a organiza toate informațiile unei comenzi online, fiecare cheie din dicționar reprezentând un tip de informație (de exemplu, „client”, „adresă”, „produse”), iar valoarea asociată acelei chei conținând detaliile respective, astfel încât toate datele comenzii să fie grupate logic și să fie ușor de găsit - 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; - explicarea modului în care un dicționar este folosit pentru a structura datele într-un sistem de gestiune a proiectelor, fiecare cheie reprezentând numele unei echipe, iar valoarea asociată fiind lista sarcinilor acelei echipe, ceea ce permite vizualizarea clară și separată a responsabilităților fiecărei echipe - explicarea modului în care structura unui orar poate fi reprezentată printr-un dicționar, în care fiecare cheie corespunde unei zile, iar valoarea asociată este lista orelor din acea zi, ceea ce permite accesul rapid la programul complet al unei anumite zile	

Clasa a X-a	
- explicarea caracteristicilor unui tuplu în contextul unei aplicații financiare, evidențiind motivul pentru care tuplurile sunt folosite pentru informații fixe (precum perechi de tipul „nume client – CNP”), astfel încât datele care nu trebuie schimbate să rămână protejate - explicarea modului în care tuplurile sunt adecvate pentru stocarea înregistrărilor fixe într-o aplicație de calendar, unde fiecare eveniment este reprezentat printr-un tuplu (data, ora, titlu_eveniment), ceea ce oferă o structură constantă și ușor de prelucrat pentru toate intrările	
CS 2.2. Explică etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea listelor, a șirurilor de caractere	
- 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 - 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, identificând eventualele suprapuneri) - 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 - explicarea modului de funcționare al diferitelor metode de criptare (de exemplu, cifrul lui Cezar, substituție simplă, cifrul Vigenère) în contextul protejării unui mesaj trimis online, evidențiind cum transformările aplicate textului urmăresc ascunderea semnificației lui - explicarea modului în care diferitele metode de criptare produc modificări specifice asupra unui șir de caractere, în contextul mesajelor codificate din jocuri de tip „vânătoare de comori”, evidențiind faptul că literele pot fi deplasate, înlocuite sau reordonate în funcție de metoda utilizată - descrierea modului în care metodele simple de criptare se aplică pe exemple concrete, de exemplu, folosirea cifrului lui Cezar pentru a transforma cuvântul „ANA” în „DQC” prin deplasarea fiecărei litere cu 3 poziții în alfabet, utilizarea unui cifru prin substituție pentru a codifica cuvântul „CARTE” folosind un tabel în care $C \rightarrow M$, $A \rightarrow Q$, $R \rightarrow T$, $T \rightarrow A$, $E \rightarrow Z$, respectiv aplicarea cifrului Vigenère pentru a cripta un mesaj cu un cuvânt cheie, obținând un șir în care fiecare literă este modificată adunând numărul de ordine din alfabet al literei din mesaj cu numărul de ordine al caracterului din cheia de criptare modulo 26 - explicarea etapelor algoritmilor de umplere (Flood Fill), arătând cum pot fi folosiți pentru a determina câte insule există pe o hartă reprezentată prin linii și coloane de valori 0 și 1, unde 1 indică uscat și 0 apă	
CS 2.3. Explică etapele unor strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor	
- explicarea etapelor metodei Divide et impera (descompunere, rezolvare, combinare), folosind ca exemplu calcularea sumei unor numere dintr-o listă, prin împărțirea repetată a acesteia și a fiecărei liste obținute pe parcurs în jumătăți (până când se ajunge la liste formate dintr-un singur număr, pentru care se cunoaște rezultatul) și determinarea rezultatului pentru o astfel de listă, prin adunarea sumelor corespunzătoare celor două jumătăți, după obținerea acestora - explicarea modului în care se aplică metoda Divide et impera, descriind clar cele trei etape ale sale — descompunerea problemei în subprobleme mai mici, rezolvarea fiecărei subprobleme, apoi combinarea rezultatelor în exemple reale în care această abordare este folosită, precum procesarea imaginilor sau navigația GPS - 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 - 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 $3+3$) - 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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice	
- explicarea modalității de parcurgere a elementelor unei mulțimi în Python prin analogie cu procesul de verificare a fiecărui obiect dintr-o cutie care conține piese unice, fără a ține cont de ordinea lor, evidențiind modul în care o instrucțiune repetitivă permite accesarea fiecărui element al mulțimii - explicarea efectului aplicării metodelor (add(), remove(), union(), intersection()) și operatorilor (, &, -) asupra a două mulțimi în Python, în contextul ilustrării în limbaj de programare a unor exemple inspirate din viața cotidiană (de exemplu,	

Clasa a X-a

compararea listelor de prieteni din două rețele sociale, respectiv a prietenilor existenți în oricare rețea, a prietenilor comuni, a prietenilor prezenți doar în una dintre rețele)

- *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())*
- *explicarea efectului aplicării unor metode specifice dicționarilor prin asocierea acestora cu gestionarea informațiilor din agenda telefonică, de exemplu, verificarea faptului că un nume se află în agendă (get()), obținerea listei care conține doar numele din agendă (keys()), obținerea listei care conține doar numerele de telefon din agendă (values()), obținerea listei complete nume-număr din agendă (items())*
- *explicarea rezultatelor obținute în urma iterării asupra unui dicționar, cu exemplificare pe agenda telefonică, din care, în urma iterării, se poate obține doar lista de nume, doar lista de numere de telefon sau lista de perechi nume-număr de telefon, în funcție de metoda de parcurgere utilizată*
- *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*
- *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*
- *explicarea efectului aplicării funcțiilor uzuale pentru prelucrarea tuplurilor în contextul memorării numărului de ore pentru activități extrașcolare din fiecare zi a săptămânii (de exemplu, ore_pe_zi = (3, 2, 4, 1, 5, 2, 0)) și aflarea numărului total de zile cu activități extrașcolare (len()), a celei mai aglomerate zile (max()), a zilei cu cele mai puține ore (min()), a numărului total de ore (sum()), a orelor în ordine crescătoare (sorted()), verificarea faptului că există cel puțin o zi cu ore programate (any()) sau că toate zilele reținute au ore programate (all())*
- *explicarea operației de extragere a elementelor unui tuplu în variabile (despachetare), în contextul salvării numărului de ore pentru activități extrașcolare în variabile aferente fiecărei zile din săptămână (de exemplu, dacă ore_pe_zi = (3, 2, 4, 1, 5, 2, 0), putem salva numărul de ore din fiecare zi astfel: luni, marti, miercuri, joi, vineri, sambata, duminica = ore_pe_zi)*
- *explicarea efectului utilizării operatorilor pentru combinarea (+) și extinderea tuplurilor (*), în contextul combinării echipamentelor sportive pentru două echipe sau a extinderii unui tuplu care reprezintă meniul comandat de mai multe persoane într-o cantină*
- *explicarea modului în care metodele clasei string din C++ permit căutarea și copierea unui șir de caractere în altul, folosind ca exemplu salvarea într-o variabilă separată a adresei de e-mail introduse de un utilizator în cadrul unui text, pentru a fi folosită ulterior la trimiterea unei notificări*
- *explicarea modului în care poate fi accesat un caracter individual dintr-un șir folosind indexarea în C++, în contextul extragerii primei litere dintr-un nume introdus într-un formular pentru a stabili dacă începe cu o literă specifică (de exemplu, pentru gruparea elevilor după inițiala numelui)*
- *explicarea modului în care o structură din C++ poate fi definită, pentru a reuni date de tipuri diferite, folosind ca exemplu organizarea informațiilor despre un elev (nume, prenume, medie, număr matricol) într-o formă unitară, similar modului în care acestea sunt grupate într-un formular școlar, sau a informațiilor legate de un produs dintr-un magazin online (de exemplu, denumire, preț, cantitate în stoc), evidențiind motivul pentru care datele sunt grupate și nu tratate separat*
- *explicarea modului în care se accesează și se utilizează câmpurile unei structuri în C++, folosind ca exemplu o structură care stochează detalii despre o rezervare la hotel (nume client, număr de nopți, tipul camerei, preț pe nopți), pentru a arăta cum fiecare informație poate fi preluată și folosită atunci când se calculează costul total pentru un client*

CS 2.5. Explică mecanismul de executare a subprogramelor recursive, pentru a le utiliza în implementarea algoritmilor în limbaj de programare

- *explicarea modului de executare a unui subprogram recursiv, descriind cum fiecare apel este plasat pe stivă și rămâne în așteptare până la rezolvarea apelului următor, folosind ca exemplu o serie de cutii puse una în alta: întâi se deschide cutia cea mai mare, apoi cutia aflată în interiorul ei și tot așa până la cutia cea mai mică; abia după ce se ajunge la ultima cutie și se preia obiectul plasat în aceasta, cutiile sunt închise în ordinea inversă deschiderii, exact cum apelurile recursive sunt rezolvate în ordine inversă plasării lor pe stivă*

Clasa a X-a

- explicarea modului în care un algoritm iterativ poate fi transformat într-o variantă recursivă echivalentă, folosind ca exemplu calcularea sumei primelor n numere naturale: varianta iterativă adună pe rând numerele de la 1 la n într-un ciclu, iar varianta recursivă calculează suma exprimând-o ca valoarea curentă plus suma numerelor rămase
- explicarea modului în care apelul recursiv al unui subprogram conduce la reluarea instrucțiunilor acestuia, într-un alt context, mai restrâns, pentru rezolvarea unor probleme matematice precum calcularea factorialului, determinarea celui de-al n -lea termen al unui șir definit recursiv sau calculul celui mai mare divizor comun prin scăderi repetate, evidențiind necesitatea unui caz de oprire și a faptului că funcția se apelează repetat

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 - neliniare, liniare, asociative, mixte, pentru a rezolva probleme de gestionare a datelor**

- aplicarea operațiilor de reuniune și intersecție asupra a două mulțimi de clienți din două magazine online, pentru a stabili atât lista tuturor clienților care pot primi oferte promoționale, cât și clienții comuni celor două magazine
- utilizarea mulțimilor pentru generarea elementelor unice (de exemplu, coduri de acces fără repetiție), verificarea dublurilor într-o listă de elemente (de exemplu, produse identice într-un inventar), verificarea faptului că o colecție conține toate valorile dintr-o secvență de elemente (de exemplu, verificarea faptului că toate zilele unei săptămâni au fost completate într-un raport)
- utilizarea comparării lexicografice pentru ordonarea titlurilor de cărți într-o bibliotecă digitală, astfel încât acestea să fie afișate alfabetic într-o listă
- 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)
- 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
- utilizarea organizării datelor sub formă de dicționar pentru realizarea unui glosar online care asociază niște termeni din tehnologia informației (chei) cu explicațiile lor (valori), facilitând învățarea rapidă a vocabularului de specialitate
- utilizarea dicționarului pentru organizarea perechilor cheie-valoare și obținerea unor rezultate, în contextul parcurgerii listei de produse dintr-un magazin unde fiecare produs (cheia) are asociat un preț (valoarea) și obținerea de rezultate precum valoarea totală a stocului, prețul minim/maxim, media prețurilor, produsele cu prețul mai mare/mai mic decât o valoare dată etc.
- accesarea unui element dintr-un dicționar utilizat într-o aplicație de comerț online, în care codul unic al produsului este folosit drept cheie, iar valorile asociate sunt informațiile despre produs (denumire și preț), pentru a afișa rapid detaliile corespunzătoare codului introdus de utilizator
- 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 ;
- utilizarea tuplurilor pentru reprezentarea coordonatelor geografice într-o aplicație de navigație GPS, prin memorarea fiecărui punct de interes într-un tuplu format din două valori fixe (latitudine, longitudine), astfel încât aceste perechi să poată fi folosite direct în calcule precum determinarea distanței dintre două locații
- aplicarea unui algoritm de sortare asupra unei liste de produse reprezentate prin tupluri (denumire, preț), folosind al doilea element al fiecărui tuplu pentru a ordona produsele crescător după preț
- utilizarea tuplurilor pentru calcularea mediei la o disciplină școlară, prin memorarea notelor unui elev în tupluri de forma (disciplină, notă), selectarea tuplurilor corespunzătoare acelei discipline, adunarea notelor și împărțirea sumei la numărul de note adunate pentru a obține media

CS 3.2. Aplică algoritmi specializați pe clase de probleme, pentru a fi putea utiliza în prelucrarea listelor, a șirurilor de caractere

Clasa a X-a

- 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
- 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
- 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
- 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
- aplicarea algoritmilor de criptare corespunzători metodelor studiate pentru transformarea unui mesaj obișnuit într-un mesaj criptat, folosind reguli precise de modificare a caracterelor, în scopul protejării informațiilor transmise
- exersarea unui sistem de criptare a mesajelor folosind cifrul lui Cezar, prin introducerea unui text format din litere mari și a unei chei de criptare care indică numărul de poziții cu care fiecare literă este deplasată, obținând mesajul criptat
- utilizarea metodei substituției simple pentru transmiterea de mesaje confidențiale între elevii unei clase, prin înlocuirea fiecărei litere din mesaj cu o altă literă stabilită conform unui tabel de substituție cunoscut, astfel încât mesajul rezultat să poată fi citit doar de destinatarii care cunosc regula de corespondență
- aplicarea metodei de sortare rapidă pentru ordonarea unei liste de valori date, prin alegerea unui element de referință (pivotal) și reorganizarea celorlalte elemente, obținând două subliste de elemente mai mici sau egale și respectiv mai mari decât pivotul și replicarea aceleiași strategii pentru cele două subliste, recursiv, de exemplu, pentru ordonarea efectivă a celor 10 note obținute de niște elevi la un test, pentru a afișa rezultatele de la cea mai mică la cea mai mare notă
- aplicarea algoritmilor de umplere pentru a determina suprafața unei zone continue, folosind ca exemplu un ocean identificat pe o hartă formată din puncte de uscat și apă, utilizând reprezentări matriceale ale unor suprafețe, pornind de la un punct de uscat, pentru a determina suprafața insulei (numărul de puncte de uscat) corespunzătoare

CS 3.3. Aplică strategii de tip Divide et impera, 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
- 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)
- aplicarea metodei de sortare prin interclasare pentru obținerea unei liste ordonate pornind de la o listă neordonată formată din 7 numere date
- 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
- 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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice

- utilizarea unor operatori (|, &, -) și metode (add(), remove(), union(), intersection()) ale clasei set în contexte practice, de exemplu, pentru gestionarea participării elevilor la activități extrașcolare, cu evidențierea modului în care aceste operații pot simplifica prelucrarea și compararea datelor
- utilizarea unor metode adecvate din clasa set în implementarea unor programe pentru verificarea faptului că o listă este formată din elemente distincte (de exemplu, coduri unice de participare), eliminarea elementelor duplicate dintr-o listă (de exemplu, coduri introduse de mai multe ori), determinarea elementelor comune între două colecții (de exemplu, elevii înscriși la două cercuri diferite)
- 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())
- 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())

Clasa a X-a

- aplicarea metodelor uzuale ale clasei dict în contexte practice, de exemplu, pentru gestionarea produselor dintr-un magazin: actualizarea stocului de produse (update()), ștergerea unui articol din magazin (pop()), ștergerea ultimului produs adăugat (popitem()), lichidare de stoc (clear()), crearea unei copii de siguranță a stocului înainte de modificări majore (copy()), crearea unui nou stoc pornind de la un stoc existent (fromkeys()) în care vor fi toate produsele inițiale (cheile) dar prețurile aferente (valorile) au valoarea None, verificarea existenței unui produs în magazin cu scopul obținerii prețului acestuia și evitarea erorilor în cazul în care produsul lipsește (get()), adăugarea automată a unui produs în magazin dacă nu există deja (setdefault()), obținerea listelor cu denumirile produselor pentru inventar (keys()), prețurile din magazin (values()) sau perechile produs-preț (items()) pentru afișare pe etichete
- utilizarea de dicționare imbricate, cu exemplificare pe gestionarea stocului unui magazin (reprezentat sub formă de dicționar) în care fiecare produs (cheia) are asociat un alt dicționar cu detalii ca preț, cantitate etc.
- 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
- 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
- aplicarea mecanismului de despachetare a unui tuplu (tuple unpacking) pentru extragerea valorilor (de exemplu, integrarea într-o aplicație de gestionare a elevilor, în care funcțiile returnează mai multe informații)
- aplicarea metodelor și funcțiilor uzuale pentru tupluri în rezolvarea unor probleme practice, de exemplu, pentru gestionarea unui magazin, în care încasările săptămânale sunt memorate în tupluri, și pe baza acestora se determină numărul total de zile cu încasări, ziua cu cele mai multe/puține încasări, suma totală încasată pe săptămână, încasările sortate crescător/descrescător, verificarea existenței a cel puțin unei zile cu încasări, verificarea faptului că au fost încasări în fiecare zi înregistrată
- utilizarea metodelor adecvate ale clasei string din C++ în vederea determinării numărului de caractere dintr-un text introdus de utilizator, cum ar fi un mesaj sau un nume complet, cu scopul verificării încadrării într-o limită impusă, de exemplu, lungimea maximă permisă a unui mesaj
- utilizarea metodei clasei string din C++ pentru copierea unui text într-un alt șir de caractere, de exemplu, duplicarea conținutului unui câmp „adresă” într-un formular electronic pentru salvare sau prelucrare ulterioară
- aplicarea metodelor de căutare ale clasei string din C++ pentru a determina poziția unui caracter într-un text, folosind ca exemplu verificarea existenței spațiului dintre nume și prenume într-un nume complet introdus de utilizator, prin identificarea poziției primei apariții a caracterului spațiu
- utilizarea unei structuri în C++ pentru organizarea informațiilor despre un produs dintr-un magazin, prin stabilirea câmpurilor necesare (nume, preț, cantitate), declararea variabilelor corespunzătoare și folosirea acestora pentru memorarea, accesarea și modificarea valorilor fiecărui câmp

CS 3.5. Utilizează elementele de sintaxă și componentele din definiția și apelul subprogramelor recursive, în implementarea algoritmilor în limbaj de programare

- aplicarea elementelor de sintaxă pentru implementarea unui subprogram recursiv, pe baza unei relații recurente date, care determină factorialul unui număr, de exemplu, pentru stabilirea numărului de moduri diferite în care pot fi așezate 4 persoane pe 4 scaune într-o sală
- aplicarea recursivității pentru determinarea unui termen din șirul lui Fibonacci, prin urmărirea apelurilor unui subprogram recursiv în Python
- aplicarea recursivității pentru calcularea sumei elementelor unei liste, pe baza unui subprogram recursiv dat în limbajul Python, care adună elementul curent al listei cu rezultatul obținut prin apelarea sa asupra restului listei
- implementarea unui subprogram recursiv pornind de la o relație de recurență dată, de exemplu, pentru calculul combinațiilor de câte k elemente dintre cele n date, pe baza formulei $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ cu respectarea condițiilor matematice
- aplicarea recursivității pentru prelucrarea cifrelor unui număr întreg, prin urmărirea pas cu pas a unui subprogram în C++ care extrage o cifră la fiecare apel și se autoapelează pentru numărul obținut prin eliminarea acesteia, până la epuizarea tuturor cifrelor, pentru a calcula suma, produsul sau media aritmetică a cifrelor selectate

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 - neliniare, liniare, asociative, mixte, pentru a alege structura adecvată în rezolvarea unor probleme concrete	
- <i>analizarea avantajelor și limitărilor utilizării mulțimilor în comparație cu listele, prin raportare la situații practice, precum verificarea rapidă a înscrierii unui elev în catalogul clasei, cu evidențierea faptului că verificarea apartenenței unui element este mai eficientă din punctul de vedere al numărului de operații efectuate, utilizând mulțimi comparativ cu liste, dar acestea nu păstrează ordinea și nu acceptă elemente duplicate</i> - <i>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</i> - <i>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)</i> - <i>analizarea avantajelor utilizării unui dicționar într-o aplicație financiară, în care codul fiscal al fiecărui client este utilizat drept cheie pentru accesarea directă a datelor asociate, astfel încât informațiile despre un client sunt obținute imediat, fără a fi nevoie de verificarea succesivă a tuturor celorlalți clienți din evidență</i> - <i>analizarea comportamentului datelor organizate în dicționare la actualizarea valorilor pentru chei existente versus chei noi, în contextul modificării prețului unui produs existent (cheie existentă) sau la adăugarea unui produs nou (cheie nouă) într-un magazin</i> - <i>analizarea modului în care este utilizat un dicționar într-un sistem de evidență al unei biblioteci, în care fiecare carte este identificată printr-o cheie (de exemplu, codul cărții), iar valoarea asociată indică plasarea acesteia, pentru a verifica dacă fiecărei chei îi corespunde o plasare, evitând astfel pierderea informației despre plasarea cărților în bibliotecă</i> - <i>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</i> - <i>analizarea adecvării utilizării tuplurilor, pentru a organiza datele (disciplinele) în cadrul unui program săptămânal fixat</i> - <i>examinarea modului în care folosirea tuplurilor într-o aplicație medicală contribuie la păstrarea neschimbată a datelor esențiale ale unui pacient (nume, CNP, data nașterii), prin utilizarea unei structuri cu valori fixe care limitează alterarea accidentală a informațiilor</i> - <i>analizarea modului în care perechile (produs, preț) sunt memorate sub formă de tupluri într-un sistem de vânzări, pentru a verifica dacă prețul asociat fiecărui produs este corect utilizat în prelucrări precum afișarea costului unui articol, calculul valorii totale a unei comenzi sau compararea prețurilor între produse</i>	
CS 4.2. Analizează comportamentul, aplicabilitatea, avantajele și dezavantajele utilizării unor algoritmi specializați pe clase de probleme pentru prelucrarea listelor, a șirurilor de caractere	
- <i>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</i> - <i>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</i> - <i>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</i> - <i>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</i> - <i>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</i> - <i>compararea metodelor de criptare studiate (cifrul lui Cezar, substituția simplă, cifrul Vigenère) din perspectiva modului de transformare a mesajului, a tipului de cheie utilizate și a nivelului de dificultate în descifrarea unui mesaj real, precum transmiterea unui mesaj confidențial între două persoane</i> - <i>analizarea mesajelor criptate obținute în situații concrete cu metodele studiate, precum trimiterea unui cod de acces sau a unui mesaj privat, pentru a identifica erori de criptare, cum ar fi litere transformate incorect, pierderea unor caractere sau aplicarea greșită a regulilor metodei alese</i>	

Clasa a X-a
- examinarea asemănărilor și deosebirilor dintre metodele de criptare utilizate în contexte reale, precum comunicarea mesajelor secrete sau protejarea informațiilor scrise, prin raportare la modul de transformare a literelor, dependența de cheie și ușurința cu care mesajul poate fi descifrat - analizarea aplicării metodei de sortare rapidă într-un context real, precum ordonarea comenzilor unui magazin după valoare, prin urmărirea pașilor algoritmului - analizarea modului de aplicare a algoritmilor de umplere într-un context precum identificarea unei zone continue de teren pe o hartă digitală, prin pornirea dintr-un punct inițial - compararea algoritmilor de sortare prin interclasare și sortare rapidă din perspectiva numărului de pași necesari și a timpului de rezolvare, în contexte precum ordonarea unor liste mari de valori ale unor comenzi
CS 4.3. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor
- analizarea aplicării metodei Divide et impera printr-un exemplu concret, în care o listă de scoruri ale jucătorilor este împărțită repetat în jumătate, până când se ajunge la liste formate dintr-un singur scor, fiecare sublistă este prelucrată separat prin aceeași regulă (de exemplu, determinarea scorului maxim/minim din sublistă), iar rezultatele obținute sunt apoi comparate pentru a stabili scorul maxim din lista inițială, evidențiind explicit etapele de împărțire, rezolvare și combinare - analizarea procesului de împărțire succesivă a unei probleme prin metoda Divide et impera, printr-un exemplu precum calculul sumei elementelor unei liste - 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 - 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 - 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 - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice
- analizarea mesajelor de eroare apărute în Python la includerea într-o mulțime a unor elemente mutabile (de exemplu, liste sau dicționare), prin asocierea cu situația în care într-o colecție de obiecte unice se încearcă adăugarea unui obiect care își poate schimba forma (de exemplu, o trusă de geometrie ce poate conține doar reprezentări ale unor corpuri geometrice care sunt rigide, deci nu se pot modifica, comparativ cu cazul în care acestea ar fi realizate din plastilină, astfel încât să poată fi remodelate) - analizarea comportamentului unor programe care realizează operații de bază cu mulțimi (de exemplu, verificări rapide ale apartenenței, identificarea dublurilor într-o listă de înscrieri), utilizând clasa set, respectiv list, pentru alegerea unei variante eficiente din punctul de vedere al timpului de executare - analizarea avantajelor utilizării unei liste de șiruri de caractere, comparativ cu un șir compact, pentru afișarea în ordine alfabetică a unor cuvinte - 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 - analizarea avantajelor utilizării dicționarelor în contextul comparării stocurilor a două magazine pentru a identifica produse disponibile în ambele magazine (chei comune), produse care există doar într-un magazin și nu în celălalt (diferențe) - analizarea modului în care pot fi combinate dicționarele, în contextul unificării stocurilor din mai multe magazine, pentru adăugarea sau actualizarea datelor produselor din magazinul principal (update()), unificarea stocurilor pentru generarea unui catalog de produse (operatorii “ ”, “**”), unificarea manuală și verificarea cheilor duplicate, filtrarea produselor cu preț mai mare decât o valoare dată în momentul unificării - 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

Clasa a X-a	
- analiza 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 - analiza avantajelor utilizării unui tuplu care conține obiecte mutabile (liste, dicționare), în contextul reținerii și prelucrării datelor despre produsele dintr-un magazin într-un tuplu de liste, de exemplu, magazin=([‘paine’, 20], [‘lapte’, 15], [‘mere’, 30]), unde pe prima poziție a fiecărei liste este denumirea produsului, iar pe a doua poziție este cantitatea aferentă - analiza modului în care operațiile de bază asupra șirurilor de caractere în C++(copiere, concatenare, căutare, comparare) pot fi realizate fie prin soluții construite pas cu pas, fie prin utilizarea funcționalităților oferite de clasa string, evidențiind diferențele dintre cele două abordări în contexte reale precum prelucrarea numelor utilizatorilor, validarea datelor introduse într-un formular sau generarea automată de mesaje text, din perspectiva clarității codului, riscului de erori și eficienței în utilizare - analiza modului în care o structură imbricată în C++ modelează un obiect format din mai multe componente, de exemplu, o mașină descrisă printr-o structură care include date generale (marcă, an) și o structură separată pentru motor (tip, capacitate), pentru a evidenția clar relația dintre întreg (mașina) și părțile sale (motorul), precum și modul în care informațiile fiecărei părți contribuie la descrierea completă a obiectului	
CS 4.5. Analizează aplicabilitatea, avantajele și dezavantajele utilizării subprogramele recursive, în implementarea algoritmilor în limbaj de programare	
- analiza consumului de timp și memorie al unui subprogram recursiv în Python care rezolvă o problemă matematică, precum determinarea sumei primelor n numere naturale, prin observarea numărului de apeluri recursive efectuate și a spațiului ocupat în stiva de executare pe măsură ce parametrul utilizat se modifică - compararea abordării recursive și a celei iterative în Python pentru rezolvarea unei probleme matematice precum calculul sumei primelor n numere naturale, analizând diferențele dintre cele două moduri de rezolvare în ceea ce privește numărul de pași efectuați, modul de repetare a operațiilor și utilizarea memoriei în funcție de valoarea lui n - compararea a doi algoritmi pentru determinarea unui termen al șirului lui Fibonacci, unul iterativ, iar celălalt recursiv, pe baza desfășurării pas cu pas a executării acestuia, urmărind ordinea apelurilor și revenirea din acestea, prin analogie cu verificarea succesivă a unor cutii puse una în alta, unde fiecare verificare nouă este memorată temporar până când se ajunge la ultima cutie și se revine treptat la prima, observând reluarea unor calcule - analiza modului în care utilizarea recursivității în C++ influențează timpul de executare și consumul de memorie prin acumularea apelurilor în stiva de executare (de exemplu, pentru determinarea valorii unei puteri a^n prin apeluri succesive ale aceleiași funcții, în funcție de valoarea exponentului n)	

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 - neliniare, liniare, asociative, mixte și a modurilor de prelucrare a acestora pentru rezolvarea unor probleme informatice concrete	
- argumentarea adecvării utilizării mulțimilor care gestionează date neordonate și fără elemente duplicate (de exemplu, administrarea codurilor unice ale produselor dintr-un magazin, evidența participanților la un eveniment) cu justificarea alegerii acestora pentru avantajele de unicitate - evaluarea avantajelor utilizării mulțimilor în organizarea unui volum mare de informații despre clienți, în comparație cu alte forme de organizare precum listele, justificând alegerea mulțimilor prin capacitatea lor de a elimina automat aparițiile repetate și de a permite identificarea rapidă a elementelor unice, ceea ce reduce erorile și timpul de prelucrare a datelor - 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 - 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 - evaluarea corectitudinii utilizării unui dicționar digital într-o aplicație reală (de exemplu, un catalog electronic de produse), prin verificarea faptului că fiecare cheie este unică și că valoarea asociată conține informații complete și corecte, justificând că această structură previne confuziile și erorile de afișare în timpul utilizării - justificarea alegerii dicționarului pentru rezolvarea unor probleme din context cotidian (de exemplu, gestionarea unui catalog de elevi, a unei biblioteci, a unui concurs sportiv, a unei agenții de turism, organizarea informațiilor economice	

Clasa a X-a

cum ar fi asocierea codurilor de factură cu datele corespunzătoare ale clienților), prin accesul rapid la date și prin claritatea relației dintre identificator și informația asociată

- *evaluarea avantajelor utilizării unei structuri de tip dicționar care asociază termeni cu definițiile lor (de exemplu, un glosar digital), prin aprecierea corectitudinii relației termen–definiție și justificarea modului în care această organizare sprijină regăsirea rapidă și fără ambiguități a informațiilor*
- *evaluarea corectitudinii datelor într-un model conceptual mixt (de exemplu, listă de liste, listă de dicționare) 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)*
- *argumentarea alegerii unui model mixt pentru un sistem de catalog electronic (de exemplu, listă de liste, listă de dicționare) pentru a organiza datele despre clase, elevi și note, asigurând o structură clară și flexibilă*
- *evaluarea adecvării utilizării tuplurilor pentru memorarea unor date care nu se modifică, precum coordonatele geografice ale unui punct sau datele de identificare ale unui document, justificând alegerea prin stabilitatea valorilor și reducerea riscului de modificări accidentale*
- *justificarea alegerii unui tuplu în locul unei liste pentru a organiza date constante (de exemplu, coordonatele unui punct fix pe hartă, programul fix al săptămânii, coduri de culori RGB etc.)*

CS 5.2. Evaluează corectitudinea și eficiența soluțiilor cu algoritmi specializați pe clase de probleme pentru prelucrarea listelor, a șirurilor de caractere

- *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*
- *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*
- *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*
- *evaluarea adecvării metodelor de criptare și decriptare utilizate într-un context real, precum protejarea mesajelor transmise într-o aplicație de comunicare, prin justificarea alegerii metodei (cifru lui Cezar, substituție simplă, cifru Vigenère) în funcție de nivelul de securitate necesar, tipul cheii folosite și ușurința de aplicare*
- *evaluarea eficienței utilizării metodelor de criptare într-un context real de transmitere a documentelor, de exemplu, trimiterea prin e-mail a unui fișier cu date personale, prin analizarea modului în care textul este criptat cu cifru Vigenère pentru confidențialitate, în vederea justificării faptului că informația este protejată împotriva accesului neautorizat*
- *evaluarea eficienței metodei de sortare rapidă în organizarea unui volum mare de date (de exemplu, ordonarea notelor elevilor participanți la un examen), prin justificarea alegerii pivotului, a modului de divizare a datelor și a impactului acestora asupra timpului de executare*
- *evaluarea corectitudinii și oportunității utilizării algoritmilor de umplere (Flood Fill) într-un context practic (de exemplu, colorarea unei regiuni delimitate pe o hartă), prin justificarea respectării condițiilor de vecinătate, a limitelor de extindere și a acoperirii exacte a zonei vizate, fără a afecta elementele exterioare*

CS 5.3. Evaluează corectitudinea și eficiența algoritmilor care utilizează strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor

- *evaluarea eficienței metodei Divide et impera în rezolvarea unor probleme cotidiene (de exemplu, sortarea unui șir de melodii după durată sau căutarea rapidă a unui artist într-o listă ordonată alfabetic), prin justificarea alegerii metodei pe baza timpului de executare, a clarității pașilor și a comparării cu alte metode de rezolvare*
- *evaluarea oportunității aplicării metodei Divide et impera în contexte matematice (de exemplu, căutarea unui element într-un interval numeric prin înjumătățiri succesive), prin justificarea deciziei pe baza modului de împărțire a problemei în subprobleme independente, a reducerii numărului de pași și a eficienței soluției finale*
- *evaluarea corectitudinii unui algoritm bazat pe metoda Divide et impera utilizat într-o problemă matematică (de exemplu, calculul rapid al unei puteri sau al maximumului dintr-un șir de valori), prin justificarea validității soluției obținute pentru toate subproblemele și a modului în care acestea sunt combinate*
- *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*
- *evaluarea obținerii soluției optime pentru probleme rezolvate cu metoda Greedy (de exemplu, organizarea unui program de întâlniri fără suprapuneri), prin justificarea corectitudinii condițiilor alese*
- *argumentarea deciziei de aplicare a metodei Greedy în contexte matematice (de exemplu, realizarea unei sume maxime cu n termeni de forma $a_i \cdot b_j$ unde $a_i \in A$, $b_j \in B$, iar A și B sunt mulțimi de numere întregi, $\text{card}(A)=n$ și $\text{card}(B) \geq n$), cu justificarea eficienței obținute ca urmare a luării deciziilor locale și a respectării condițiilor de aplicabilitate ale metodei*

Clasa a X-a**CS 5.4. Evaluează corectitudinea și eficiența aplicațiilor care utilizează elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele simple - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice**

- evaluarea corectitudinii și eficienței diferitelor soluții de eliminare a elementelor duplicate dintr-o listă, prin compararea implementărilor iterative cu cele bazate pe conversia listei într-o mulțime (set), pentru a evidenția avantajele utilizării mulțimilor în simplificarea și accelerarea procesului de filtrare
- evaluarea eficienței a două soluții pentru aceeași problemă, una care integrează operații specifice clasei set pentru îmbunătățirea clarității și performanței și alta bazată pe liste (de exemplu, gestionarea datelor de înscriere la un eveniment, filtrarea elementelor unice dintr-un fișier)
- evaluarea eficienței și clarității redactării a două soluții pentru aceeași problemă, care verifică dacă un cuvânt este anagramă a unui alt cuvânt
- argumentarea alegerii unei metode din clasa str pentru o sarcină practică, de exemplu, căutarea unor vocale într-un text dat
- evaluarea eficienței unei soluții bazate pe dicționar față de una bazată pe listă de liste (listă de perechi), în contextul căutării rapide a prețului unui produs într-un catalog
- evaluarea unui cod care folosește inadecvat metodele clasei dict (de exemplu, încercarea de a accesa o cheie inexistentă fără metoda get(), utilizarea metodei pop() cu argument greșit, încercarea de a modifica o cheie etc.)
- 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
- evaluarea corectitudinii declarării unei variabile de tip tablou bidimensional în C++ din punctul de vedere al necesarului de memorie utilizată, pentru memorarea 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
- evaluarea clarității unui cod care folosește tupluri pentru transmiterea mai multor valori dintr-o funcție (de exemplu, returnarea simultană a datelor despre un produs dintr-un magazin, cum ar fi denumire, preț, cantitate)
- evaluarea corectitudinii unui cod cod în care sunt utilizate tupluri, având în vedere enunțul problemei corespunzătoare și corectarea eventualelor greșeli
- evaluarea corectitudinii unui program C++ care prelucrează texte din viața reală (de exemplu, validarea și modificarea unui nume de utilizator sau a unui mesaj), prin justificarea utilizării corecte a operațiilor și metodelor clasei string pentru concatenare, acces la caractere, comparare, inserare, ștergere și căutare de secvențe de caractere, pe baza rezultatelor obținute pentru diferite date de intrare
- evaluarea eficienței utilizării unei structuri în C++ pentru gestionarea datelor despre elevi (de exemplu, nume, vârstă, medie), prin justificarea clarității codului, a grupării logice a câmpurilor și a ușurinței de acces și modificare a informațiilor comparativ cu utilizarea unor variabile separate
- evaluarea oportunității folosirii unui tip de date structurat pentru stocarea și prelucrarea unui set de date din viața reală (de exemplu, evidența clienților sau a produselor), prin justificarea modului în care structura permite acces coerent la câmpuri, atribuire unitară și reducerea erorilor în raport cu utilizarea altor tipuri de date C++

CS 5.5. Evaluează corectitudinea și eficiența programelor care utilizează subprograme recursive, pentru a rezolva probleme informatice

- evaluarea eficienței unui subprogram recursiv în Python în raport cu o implementare iterativă echivalentă pentru o problemă matematică (de exemplu, calculul factorialului unui număr), prin justificarea diferențelor de claritate a soluției, număr de pași de executare și consum de memorie
- evaluarea limitărilor în utilizarea unui subprogram recursiv în Python pentru rezolvarea unei probleme matematice (de exemplu, determinarea celui de al n-lea termen al șirului lui Fibonacci), evidențiind apeluri repetate pentru realizarea unor calcule identice
- evaluarea deciziei de folosire a unui subprogram recursiv în Python pentru organizarea unui buget complex (de exemplu, calcularea soldului lunar pentru un buget structurat ierarhic, folosind liste de liste sau dicționare de dicționare), prin justificarea modului în care structura recursivă simplifică tratarea nivelurilor multiple de date față de o soluție iterativă
- evaluarea a două implementări recursive în C++ ale aceluiași algoritm matematic (de exemplu, calculul celui mai mare divizor comun), prin justificarea alegerii variantei considerate mai eficientă și mai clară pe baza structurii apelurilor recursive, a condiției de oprire și a consumului de resurse

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 - neliniare, liniare, asociative, mixte, pentru organizarea și prelucrarea eficientă a datelor, utilizând algoritmi de bază, în vederea rezolvării unor probleme	
- proiectarea unui algoritm de păstrare a evidențelor pentru o organizație educațională (de exemplu, o platformă de cursuri), pentru a identifica participanții înscriși, a evita înscrierile redundante, pe baza datelor utilizatorilor care sunt organizate în mulțimi - proiectarea unui algoritm pentru analizarea datelor unui magazin online, în care sunt definite explicit mulțimea tuturor clienților înregistrați și, pentru fiecare produs, mulțimea clienților care l-au achiziționat, iar relațiile dintre mulțimi sunt utilizate pentru recomandarea de perechi de produse care sunt achiziționate de obicei împreună - 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 - 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 - proiectarea unui algoritm de actualizare automată a unui dicționar de cursuri valutare, în care cheia este codul monedei (de exemplu, EUR, USD), iar valoarea este cursul de schimb asociat în lei, algoritmul preluând periodic date noi și înlocuind valorile existente cu cele actualizate - proiectarea unui sistem de înlocuire automată a termenilor dintr-un text, folosind un dicționar în care cheia este cuvântul sau expresia căutată, iar valoarea este cuvântul sau expresia care o înlocuiește, algoritmul parcurgând textul și substituind fiecare apariție a cheilor cu valorile corespunzătoare - proiectarea unui algoritm care utilizează un model mixt (listă de liste sau dicționar 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 - proiectarea unui algoritm de filtrare a comenzilor dintr-o listă în care pentru fiecare înregistrare se păstrează suma plătită și numărul bonului, pentru a afișa doar comenzile care depășesc o sumă prestabilită - proiectarea unui algoritm bazat pe tupluri pentru afișarea clasamentelor sportive, în care fiecare tuplu conține numele jucătorului, punctajul obținut și poziția finală, astfel încât datele să corespundă exact rezultatelor stabilite la finalul competiției și să nu poată fi modificate ulterior, păstrând coerența datelor pe toată durata competiției - proiectarea unui algoritm pentru înregistrarea itinerariului parcurs de un grup de turiști, utilizând tupluri, fiecare conținând coordonatele GPS și denumirea obiectivului vizitat	
CS 6.2. Proiectează soluții cu aplicarea personalizată a algoritmilor specializați pe clase de probleme pentru prelucrarea listelor, a șirurilor de caractere	
- 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 - 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 - 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 - proiectarea unor soluții informatice pentru protejarea mesajelor transmise într-un context real (de exemplu, comunicarea dintre membrii unei echipe sau stocarea unor note personale), prin proiectarea unor aplicații care implementează criptarea și decriptarea șirurilor de caractere folosind cifrul lui Cezar sau cifrul Vigenère, explicând modul în care cheia și regulile de transformare asigură confidențialitatea informației - proiectarea unor soluții pentru verificarea integrității datelor textuale (de exemplu, validarea unui fișier trimis online sau detectarea unor modificări unui mesaj), prin aplicarea algoritmului lui Fletcher, astfel încât soluția să fie mai eficientă și mai sigură în raport cu problema reală rezolvată - crearea unei aplicații pentru gestionarea comenzilor unui magazin online, în care lista comenzilor este sortată folosind metoda de sortare rapidă prin selectarea unui element pivot, separarea comenzilor cu valori mai mici și mai mari decât pivotul în două subliste și ordonarea recursivă a acestora, astfel încât comenzile să fie afișate rapid și corect după criteriul ales (dată sau valoare)	

Clasa a X-a	
- proiectarea unui algoritm de tip Flood Fill pentru o aplicație de desen digital, în care utilizatorul selectează un punct dintr-o zonă închisă, algoritmul identifică zona ca un ansamblu de pixeli conectați, pe baza culorii lor, și înlocuiește culoarea tuturor acestor pixeli cu una nouă, aleasă, oprindu-se la conturul zonei	
CS 6.3. Proiectează algoritmi cu aplicarea personalizată a strategiilor de tip Divide et impera, Greedy pentru rezolvarea problemelor	
- proiectarea unor algoritmi care utilizează metoda Divide et impera, pentru contexte reale (de exemplu, căutarea unui document într-o arhivă ordonată), prin definirea adecvată a datelor și a subproblemelor, astfel încât rezultatul să fie obținut împărțind problema inițială în subprobleme mai mici de același tip - proiectarea unor algoritmi pentru crearea unor fractali simpli (de exemplu, covorul lui Sierpinski), utilizând metoda Divide et impera - proiectarea unor soluții informatice care aplică personalizat metoda Divide et impera pentru problema turnurilor din Hanoi prin reprezentarea configurației tijelor folosind structuri de date adecvate și prin afișarea etapizată a mutărilor efectuate - 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 - 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 - neliniare, liniare, asociative, mixte, în vederea realizării unor soluții funcționale și performante	
- implementarea unor aplicații personalizate care utilizează clase Python specifice pentru gestionarea mulțimilor la matematică (de exemplu, un software educațional cu titlul „Mulțimi”) sau generarea unor numere distincte pentru jocuri de tip Bingo - implementarea unor aplicații personalizate pentru prelucrarea cifrelor unui număr utilizând adecvat clasa set din Python (de exemplu, determinarea cifrelor distincte ale unui număr, formarea unui număr maxim cu cifrele distincte ale unui număr dat, determinarea cifrelor comune ale unor numere date, determinarea cifrelor care nu apar în scrierea unui număr etc.) - 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') - 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) - implementarea unor programe personalizate folosind funcționalitățile clasei dict, pentru gestiunea stocurilor dintr-un magazin - implementarea unor programe personalizate folosind funcționalitățile clasei dict, pentru traducerea automată a unor cuvinte în mai multe limbi - 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 - 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 - integrarea tuplurilor în structuri mixte, cum ar fi liste de tupluri (de exemplu, pentru gestionarea stocului de produse într-un magazin, reprezentat sub formă de listă, în care fiecare produs este reținut printr-un tuplu care conține date fixe, cum ar fi codul produsului, denumirea, prețul), dicționare în care valorile sunt tupluri (de exemplu, catalog în care cheile sunt reprezentate de numele elevilor, iar valorile sunt notele fiecărui elev, reținute în tupluri) - implementarea unor aplicații care folosesc tupluri, de exemplu, pentru gestionarea punctajelor la un concurs, unde fiecare tuplu conține (nume, scor), pentru gestionarea coordonatelor geografice pentru un explorator sau pentru gestionarea codurilor de culori RGB pentru imagini	

Clasa a X-a

- crearea unei aplicații de prelucrare în C++ a șirurilor de caractere pentru un context real precum codificarea unui text pe baza unei reguli date, în care fiecare cuvânt este transformat prin păstrarea primei și ultimei litere și inversarea ordinii literelor din interiorul cuvântului, utilizând accesul la caractere, concatenarea și metodele de inserare și ștergere din clasa string
- crearea unui program C++ care prelucrează un fișier text dintr-un context real, precum un set de e-mailuri, un document administrativ sau un jurnal de activitate, în care șirurile de caractere sunt parcurse și analizate pentru numărarea aparițiilor unui cuvânt specific (de exemplu, „confirmare” într-un document oficial) și pentru identificarea adreselor de e-mail (recunoscând structura specifică '@' și '.'), utilizând operații de comparare, căutare de secvențe de caractere și acces la caractere din clasa string pentru extragerea și organizarea informațiilor relevante
- implementarea unui algoritm pentru organizarea programului zilnic al unei persoane, în care fiecare activitate este descrisă printr-o structură C++ care conține ziua, ora de început, descrierea și durata, iar activitățile sunt reunite într-o colecție ordonată, permițând vizualizarea întregului program, parcurgerea acestuia și modificarea activităților, în același mod în care un program zilnic este planificat și ajustat în viața reală
- implementarea unui algoritm cu structuri de date imbricate în C++ pentru gestionarea evidenței unei biblioteci, în care structura Carte conține câmpuri precum titlu, autor și an, iar structura Imprumut include un câmp de tip Carte împreună cu informații suplimentare despre cititor și data împrumutului

CS 6.5. Implementează aplicații în limbaj de programare care integrează subprograme recursive personalizate, pentru a modulariza și organiza eficient codul în cadrul soluțiilor informatice

- crearea unei aplicații pentru prelucrarea cifrelor unui număr care afișează mai multe opțiuni, iar pentru fiecare opțiune se apelează câte un subprogram recursiv (de exemplu, pentru afișarea cifrelor în ordine de la stânga la dreapta, de la dreapta la stânga, determinarea cifrei maxime, a numărului de cifre, a sumei cifrelor pare, verificarea existenței unei cifre pare, verificarea proprietății de platou format cu cifre egale), fiecare subprogram fiind elaborat de câte un membru al unei echipe de elevi
- crearea unei aplicații cu subprograme recursive pentru prelucrarea unei liste ordonate de numere (de exemplu, mediile unor elevi), prin care să se afișeze numerele atât în ordine crescătoare, cât și în ordine descrescătoare
- crearea unui subprogram recursiv pentru a verifica dacă un cuvânt este o anagramă a unui alt cuvânt, în contextul unei aplicații de jocuri enigmatice
- crearea unei aplicații cu subprograme recursive care să afișeze pe ecran diverse forme geometrice simple cu ajutorul spațiilor și al unor simboluri (de exemplu, un triunghi format din simboluri *), în contextul unui concurs de realizare a designului unei sigle

CONȚINUTURI ALE ÎNVĂȚĂRII

Clasa a X-a

Domenii de conținut	Conținuturi
1. Organizarea conceptuală a datelor	1.1. Modelul conceptual neliniar - mulțime - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.2. Modelul conceptual liniar – șir de caractere - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.3. Modelul conceptual asociativ – dicționar - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.4. Modelul conceptual liniar - tuplu - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.5. Modelul conceptual mixt (de exemplu, liste de liste, liste de șiruri de caractere) - 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 - caracteristici ale metodei căutării binare și repere pentru aplicarea acesteia; - caracteristici ale metodei interclasării și repere pentru aplicarea acesteia. 2.2. Metode simple de criptare a șirurilor de caractere și verificare a integrității datelor - caracteristici ale criptării și decriptării prin metoda substituției simple (substituție monoalfabetică); - caracteristici ale criptării și decriptării folosind cifrul lui Cezar și repere pentru aplicarea metodei; - caracteristici ale criptării și decriptării utilizând cifrul Vigenère și repere pentru aplicarea metodei; - caracteristici ale sumei de control a unui șir de caractere sau a unui fișier text și determinarea sa utilizând algoritmul lui Fletcher. 2.3. Metoda Divide et impera - caracteristici ale metodei Divide et impera; - repere pentru aplicarea metodei; - caracteristici ale metodei de sortare prin interclasare și repere pentru aplicarea acesteia; - caracteristici ale metodei de sortare rapidă și repere pentru aplicarea acesteia. 2.4. Algoritmi de umplere - caracteristici ale algoritmilor de umplere (Flood Fill) și repere pentru aplicarea acestora. 2.5. Metoda Greedy - caracteristici ale metodei Greedy; - repere pentru aplicarea metodei.
3. Memorarea datelor și organizarea codului în limbaj de programare	3.1. Clasa set din Python – clasă predefinită pentru mulțimi - caracteristici; - operații și operatori specifici: reuniune, intersecție, diferență, egalitate, incluziune, apartenență; - metode uzuale pentru adăugare, ștergere, inițializare; - repere pentru identificarea și utilizarea altor metode adecvate pentru prelucrarea mulțimilor, în funcție de nevoile proprii. 3.2. Clasa str din Python – clasă predefinită pentru șiruri de caractere - caracteristici; - operații și operatori specifici: apartenență, concatenare, multiplicare, acces la un caracter sau subșir/secvență de caractere, comparare lexicografică; - 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;

Domenii de conținut	Conținuturi
	- repere pentru identificarea și utilizarea altor metode și funcții adecvate pentru prelucrarea șirurilor de caractere, în funcție de nevoile proprii. 3.3. Clasa string din C++ – clasă predefinită pentru șiruri de caractere - caracteristici; - operații și operatori specifici: concatenare, acces la un caracter, comparare lexicografică; - metode uzuale pentru inserare a unei secvențe de caractere, ștergere a unei secvențe de caractere, căutare a unei secvențe de caractere; - repere pentru identificarea și utilizarea altor metode adecvate pentru prelucrarea șirurilor de caractere, în funcție de nevoile proprii. 3.4. Clasa dict din Python – clasă predefinită pentru dicționare - caracteristici; - operații și operatori specifici: acces la date; - repere pentru identificarea și utilizarea altor metode sau operatori adecvați pentru prelucrarea dicționarelor, în funcție de nevoile proprii. 3.5. Clasa tuple din Python – clasă predefinită pentru tupluri - caracteristici; - operații și operatori specifici: acces la date, apartenență, concatenare, multiplicare; - metode uzuale pentru numărare, căutare; - repere pentru identificarea și utilizarea altor clase și metode adecvate pentru prelucrarea tuplurilor, în funcție de nevoile proprii. 3.6. Structuri de date în C++ pentru memorarea datelor neomogene (înregistrări) - caracteristici, definire a unei structuri, declarare a variabilelor de tip structură (struct); - operatori specifici pentru acces la un câmp, atribuire. 3.7. Elemente de limbaj pentru operarea cu date organizate în modele mixte - caracteristici, acces la un element în Python; - caracteristici, acces la un element în C++. 3.8. Subprograme recursive - caracteristici, rol, mecanism de executare; - repere pentru scrierea subprogramelor recursive în Python; - repere pentru scrierea subprogramelor recursive în C++.

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 măsoară? 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.

Un context de învățare poate viza simultan mai multe competențe, mai ales atunci când acestea sunt interconectate. De exemplu, pentru rezolvarea unei probleme cu ajutorul calculatorului se vizează, în cadrul aceleiași activități de învățare, atât competențe de nivel cognitiv scăzut (identificarea principalelor caracteristici ale modelelor conceptuale și operaționale implicate, explicarea principiilor care stau la baza acestora, în contextul particular al problemei, utilizarea lor în rezolvare) dar și competențe de nivel cognitiv ridicat (analizarea caracteristicilor modelelor conceptuale și operaționale specifice problemei, evaluarea corectitudinii și eficienței soluțiilor propuse și elaborarea algoritmului personalizat).

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 algoritmi 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 algoritmi 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 metode/algoritmi 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).

Pentru specializarea matematică-informatică, la clasele cu predarea disciplinei informatică în regim intensiv, **limbajele de programare pentru implementare sunt Python și C++**, de exemplu în una dintre variantele:

- în paralel (alocând pentru practică anumite ore, săptămânal, fiecărui limbaj, ca în cazul limbilor străine, sau exemplificând alternativ implementări în cele două limbaje), evidențiind pe parcurs asemănările și deosebirile dintre acestea;
- succesiv, realizându-se o recapitulare a unor concepte, în celălalt limbaj, conform programei școlare.

Predarea elementelor de limbaj specifice Python și C++ în paralel oferă o abordare solidă, pe baza unui plan didactic eficient, având în vedere consolidarea conceptelor comune și susținerea învățării prin comparație (observând asemănări de logică și structură, diferențe de sintaxă, o înțelegere mai profundă a conceptelor de bază, cum ar fi gestionarea automată sau manuală a memoriei), evidențierea complementarității conceptuale (Python oferă o sintaxă simplă, ușor de citit, care permite elevilor să se concentreze pe concepte fără să fie pus accent pe detalii tehnice, iar C++ facilitează înțelegerea profundă a mecanismelor din spatele limbajului — tipuri de date, gestionarea memoriei, pointeri, compilare), formarea unei gândiri flexibile în programare, evidențiindu-se faptul că limbajul este doar un instrument, iar gândirea logică este universală, facilitându-se astfel adaptabilitatea la orice alt limbaj viitor.

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 matematică-informatică, pentru clase cu predarea disciplinei informatică în regim intensiv, cu o abordare în paralel a celor două limbaje de programare

1. *Strategii de rezolvare a problemelor. Metode de prelucrare a listelor ordonate*
2. *Organizarea conceptuală a datelor. Modelul conceptual neliniar - mulțime*
3. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa set din Python*
4. *Organizarea conceptuală a datelor. Modelul conceptual liniar – șir de caractere*
5. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa str din Python și clasa string din C++*
6. *Strategii de rezolvare a problemelor. Metode simple de criptare a șirurilor de caractere și verificare a integrității datelor*
7. *Organizarea conceptuală a datelor. Modelul conceptual asociativ – dicționar*
8. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa dict din Python și structuri în C++*
9. *Organizarea conceptuală a datelor. Modelul conceptual liniar – tupluri*
10. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa tuple din Python*
11. *Organizarea conceptuală a datelor. Modelul conceptual mixt*
12. *Memorarea datelor și organizarea codului în limbaj de programare. Elemente de limbaj pentru operarea cu date organizate în modele mixte*
13. *Memorarea datelor și organizarea codului în limbaj de programare. Subprograme recursive*
14. *Strategii de rezolvare a problemelor. Metoda Divide et impera*
15. *Strategii de rezolvare a problemelor. Algoritmi de umplere*
16. *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/C++ 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 conceptului de mulțime reprezintă o punte naturală între gândirea matematică și cea informatică. Se recomandă utilizarea strategiei descoperirii dirijate, pornind de la situații cotidiene, de exemplu, „mulțimea elevilor din clasă”, „mulțimea culorilor preferate”, pentru a stimula înțelegerea noțiunii de element și apartenență. Se pot folosi diagrame Venn pentru reprezentarea relațiilor de reuniune, intersecție și diferență, susținând astfel dezvoltarea gândirii vizuale și logice. Se recomandă integrarea metodei comparației pentru a reliefa caracterul neliniar al acestei structuri față de listele ordonate. Profesorul concretizează teoria mulțimilor prin clasa predefinită set. Se recomandă demonstrația dirijată și învățarea activă: elevii experimentează singuri comportamentul metodelor (add, remove, union, intersection, difference). Se valorifică interdisciplinaritatea cu matematica, punând accent pe semnificația logică a operațiilor.

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/string, 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/C++ 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 limbaj de programare, pentru a exersa operațiile specifice (acces, concatenare, apartenență, căutare). Această etapă consolidează gândirea secvențială și logica parcurgerii.

Algoritmii de criptare au un rol formativ interdisciplinar, îmbinând concepte de informatică, logică, matematică și securitate a datelor. Tema oferă o introducere intuitivă în criptografie și în ideea de protejare a informației, consolidând totodată lucrul cu șiruri de caractere și prelucrări pe baza codurilor ASCII/Unicode.

Se urmărește formarea gândirii algoritmice aplicate și înțelegerea principiului de transformare a datelor (criptare ↔ decriptare), nu memorarea formulelor. Abordarea metodologică trebuie să fie intuitivă, progresivă și comparativă, pornind de la concepte simple (cifrul lui Cezar) către metode mai complexe (cifrul Vigenère), accentuând caracteristicile conceptuale și logice ale algoritmilor, principiile comune ale criptării: existența unui text clar și a unui text cifrat, utilizarea unei chei sau a unei reguli de transformare și posibilitatea de reversibilitate (criptare ↔ decriptare). Se recomandă utilizarea exemplelor concrete de mesaje și transformări simple pentru a formula regulile generale ale metodelor. Fiecare metodă se exprimă prin pași logici de prelucrare, accentuând legătura dintre idei și implementarea algoritmică și se valorifică legăturile cu alte domenii (matematică – operații modulo, statistică – frecvențe de apariție, TIC – fișiere text). Fiecare metodă poate fi analizată comparativ, în raport cu tipul transformării (substituție, deplasare, control), complexitatea algoritmului și gradul de securitate indus, evidențind asemănările și deosebirile dintre acestea (cifrul lui Cezar, cifrul Vigenère, substituție).

Profesorul poate introduce modelul asociativ, reprezentat de dicționar, prin metoda modelării, explicând principiul perechilor cheie–valoare și apelând la exemple familiare (agenda de contacte, glosar de termeni). Elevii sunt ghidați prin conversație euristică și comparare între listă și dicționar, pentru a înțelege diferențele privind accesul la date. Se recomandă integrarea live coding-ului și a vizualizării în timp real a modificării structurilor, consolidând astfel legătura între teorie și practică.

Abordarea modelului conceptual linier - tuplu poate porni de la exemple care să ilustreze ideea de colecție ordonată și imutabilă, cum ar fi coordonatele unui punct fix pe o hartă sau programul constant al unei zile. Pentru o mai bună înțelegere, se pot realiza paralele între tupluri și liste, prin analogii concrete: tuplul ca o informație permanentă (de exemplu, locația unei clădiri), iar lista ca o informație care se poate modifica (de exemplu, poziția unui robot). Compararea comportamentului tuplurilor și al listelor, din perspectiva mutabilității, performanței și siguranței datelor, evidențiind avantajele folosirii tuplurilor pentru informații fixe, poate contribui la o mai bună înțelegere a conceptelor. Se recomandă utilizarea exemplelor vizuale și contextuale (ordonate, culori RGB, programul zilnic al unei clase etc.) pentru a evidenția modul de creare, accesare și parcurgere a elementelor unui tuplu, exerciții practice care implică prelucrarea datelor reale (de exemplu, reținerea și afișarea numărului de ore de activități zilnice, combinarea tuplurilor pentru generarea unui orar extins, extragerea valorilor multiple returnate de o funcție prin despachetare). Profesorul poate valorifica situații de învățare interdisciplinare (matematică, geografie, economie) pentru a consolida înțelegerea rolului tuplurilor în stocarea datelor constante.

Predarea modelelor mixte de organizare a datelor (liste de liste, liste de șiruri de caractere, liste de dicționare etc.) poate fi abordată gradual, pornind de la exemple simple și concrete (liste de liste, liste de dicționare), în care apare nevoia de ierarhizare și acces multiplu la date, către generalizări (dicționare de dicționare). 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.

Profesorul poate introduce conceptul de recursivitate folosind metoda analogiei („oglină într-o oglindă”) și reprezentarea grafică a apelurilor recursive. Este utilă învățarea prin simulare, pentru a vizualiza mecanismul stivei de apeluri. Metacogniția are un rol major: elevii trebuie să descrie verbal condițiile de oprire și fluxul execuției.

Pentru metoda Divide et impera, profesorul poate să introducă ideea de descompunere a problemei prin problematizare și euristică. Se recomandă construirea treptată a conceptului prin exemple familiare (găsirea elementului maxim, căutarea unui element, sortarea unei liste). Diagramele arborescente și vizualizarea recursivității facilitează înțelegerea structurii logice a algoritmului. O strategie eficientă este metacogniția – elevii sunt invitați să explice verbal modul de împărțire și combinare a subproblemelor. La predare, se evidențiază aplicarea acestei strategii și pentru metoda căutării binare, cu eliminarea uneia dintre subproblemele generate, având în

vedere că nu influențează rezultatul. Se recomandă abordarea unor rezolvări cu aplicarea algoritmilor de bază (de exemplu, sume, minime), probleme clasice (de exemplu, problema turnurilor din Hanoi, problema covorului lui Sierpinski) ca model, apoi realizarea unor probleme asemănătoare care presupun adaptări, reinterprețări, extinderi.

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, reinterprețări, extinderi.

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,

Nume și prenume	Grad didactic/Titlu științific	Instituție de apartenență, localitate, județ
		Facultatea de Automatică și Calculatoare
NIȚU Alina	profesor, gradul didactic I	Liceul Teoretic „Ovidius”, Constanța, județul Constanța
NURLA Aidan	profesor, gradul didactic I	Colegiul Național Militar „Alexandru Ioan Cuza”, Constanța, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forțelor 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 Național Militar „Dimitrie Cantemir”, Breaza, județul Prahova
TEGLAȘ Maria	profesor, gradul didactic II	Colegiul Național 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

Anexa nr. 10

Programa școlară
pentru disciplina

Informatică

Clasa a XI-a

Curriculum de specialitate (CS)

*Filiera teoretică, profilul real, specializarea matematică-informatică,
clase cu predarea disciplinei informatică în regim intensiv*

Învățământ liceal

- 2026 –

Anexa nr. 10**Programa școlară pentru disciplina Informatică, clasa a XI-a, curriculum de specialitate (CS)**

- *Filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei informatică în regim intensiv*

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 matematică - informatică, clase cu predarea disciplinei *informatică* în regim intensiv, conform O.M.E.C. nr. 6873/2025 alocarea orară este:

- clasa a IX-a – 4 ore/săptămână, dintre care două ore pentru studiu teoretic și două ore pentru activități practice;
- clasa a X-a – 4 ore/săptămână, dintre care două ore pentru studiu teoretic și două ore pentru activități practice;
- clasa a XI-a – 7 ore/săptămână, dintre care patru ore pentru studiu teoretic și trei ore pentru activități practice;
- clasa a XII-a – 7 ore/săptămână, dintre care patru ore pentru studiu teoretic și trei ore pentru activități practice.

Activitățile practice sunt desfășurate obligatoriu în laboratorul de informatică, iar în baza Ordinului ministrului educației și cercetării nr. 6873/2025, pentru filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei *informatică* în regim intensiv, activitățile practice se desfășoară cu colectivul de elevi organizat pe grupe.

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.

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 limbajelor de programare Python și 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. Limbajul 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 XI-a	
CS 1.1. Identifică principalele caracteristici ale organizării datelor în cadrul unor modele conceptuale complexe – liniare, rețea, ierarhice, obiectuale, pentru a structura și accesa date în vederea prelucrării acestora	
- recunoașterea unor caracteristici ale tipurilor de liste înlanțuite (simplă, circulară), pentru exemplificarea evidenței cronologice a activităților dintr-un proiect școlar, unde o listă simplă reflectă succesiunea activităților, iar una circulară asigură revenirea la prima activitate pentru verificare - identificarea caracteristicilor unei liste dublu înlanțuite, pentru reprezentarea traseului de deplasare a autobuzelor, între locul unde sunt garate și stația finală, permițând parcurgerea stațiilor în ambele sensuri - definirea conceptului de graf, cu exemplificarea reprezentării rutelor directe bidirecționale dintre anumite localități aflate pe harta rutieră a unei țări, pentru un graf neorientat, respectiv cu exemplificarea reprezentării străzilor cu sens unic sau cu circulație pe ambele sensuri, aflate pe harta rutieră a unei localități, pentru un graf orientat - recunoașterea caracteristicilor grafurilor ponderate (orientate sau neorientate), prin completarea unei fișe comparative, pentru exemplificarea reprezentării unei hărți rutiere a unui oraș care are și străzi unidirecționale, respectiv a unei regiuni cu șosele bidirecționale, în care costurile arcelor/muchiilor indică distanța sau timpul de deplasare între intersecțiile/localitățile corespunzătoare - definirea conceptului de graf bipartit, pentru exemplificarea colorării nodurilor în roșu și negru, astfel încât oricare două noduri vecine să fie colorate diferit - identificarea unor noduri într-un arbore cu rădăcină (nod descendent direct, nod descendent, nod ascendent direct, nod ascendent, nod frate) pentru reprezentarea relațiilor ierarhice într-o organizație - definirea noțiunii de heap, pentru exemplificarea organizării ierarhice a unui sistem de priorități pentru alocarea resurselor, structura permițând accesul rapid la resursa cu prioritate maximă/minimă - identificarea caracteristicilor unei clase, prin exemplul stabilirii membrilor clasei „Rețea”, care încapsulează datele tehnice și metodele de întreținere a unei rețele - identificarea caracteristicilor moștenirii, punând în evidență „specializarea” unei clase, de exemplu pentru clasele „Autoturism” și „Camion”, care moștenesc clasa „Vehicul”	
CS 1.2. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea grafurilor, arborilor	
- precizarea etapelor algoritmului de parcurgere în lățime (BFS), în contextul explorării unei rețele de comunicații, unde algoritmul simulează verificarea tuturor legăturilor de pe același nivel - precizarea etapelor algoritmului de parcurgere în adâncime (DFS), în contextul analizării accesibilității nodurilor unei rețele de comunicații, pe care algoritmul le explorează în profunzime - definirea scopului algoritmului lui Prim, indicând tipul de graf pentru care se aplică, în contextul utilizării acestuia pentru analizarea asfaltării unor străzi dintr-un oraș, astfel încât, cu un cost minim de asfaltare, să existe posibilitatea deplasării pe asfalt între oricare două intersecții, eventual trecând pe asfalt și prin alte intersecții - precizarea scopului algoritmilor Roy-Warshall, Roy-Floyd și Kruskal, în contextul utilizării acestor algoritmi pentru analizarea rețelelor de transport între depozite (Roy-Warshall pentru drumuri între oricare două depozite, Roy-Floyd pentru drumuri minime între oricare două depozite, Kruskal pentru optimizarea conexiunilor între depozite) - enumerarea modalităților de parcurgere a arborilor binari: preordine și postordine, în contextul unei structuri ierarhice, în care fiecare angajat are cel mult doi subalterni, unde parcurgerea în preordine reflectă transmiterea mesajelor de la cel care conduce spre subordonați, iar postordine – raportarea rezultatelor de jos în sus - precizarea etapelor operațiilor specifice arborelui binar de căutare, în contextul inserării și căutării unor produse dintr-un magazin, în care fiecare produs este identificat printr-un cod unic, iar pentru gestiunea acestora se folosește un arbore binar de căutare	
CS 1.3. Identifică specificul, caracteristicile și etapele unor strategii de tip Backtracking, Programare dinamică pentru rezolvarea problemelor	
- precizarea caracteristicilor metodei de programare Backtracking, în contextul modelării procesului decizional pentru construirea unui robot, în limita unui buget dat, unde se explorează toate opțiunile posibile de alegere a pieselor necesare,	

Clasa a XI-a	
cu testarea condițiilor de continuare și eliminarea ramurilor invalide, revenindu-se la starea anterioară în momentul în care pentru starea curentă au fost explorate toate variantele - identificarea caracteristicilor metodei Backtracking, în contextul generării unor liste de obiective turistice care să fie vizitate succesiv într-un traseu turistic, respectând anumite restricții (durată, conexiuni rutiere etc.) - identificarea principiului de generare sistematică a secvențelor de valori în cadrul metodei Backtracking, în contextul testării succesive a tuturor variantelor de amplasare a unor camere web de observare pe unii dintre stâlpii unei străzi pentru acoperirea completă a acesteia - identificarea particularităților unei soluții pentru o problemă rezolvată cu metoda Backtracking generalizat, în contextul procesului de identificare a unui traseu într-un labirint, unde fiecare element al soluției este o pereche de valori (linie și coloană) - recunoașterea principiului metodei Programării dinamice, în contextul planificării meniului pentru sportivi de performanță, verificând dacă se poate obține o anumită cantitate totală de calorii pe baza unor produse date, evidențiind că problema generală este împărțită în subprobleme suprapuse și soluțiile acestora sunt reutilizate pentru a obține o soluție globală eficientă	
CS 1.4. Identifică principalele elemente ale limbajului de programare utilizate pentru memorarea și prelucrarea datelor organizate în modele complexe - liniare, rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice, cu alocare statică și dinamică a memoriei	
- definirea termenilor pointer, adresă de memorie și alocare dinamică în C++, în contextul unei analogii cu procesul de alocare a unor locuințe de serviciu, aflate la diferite adrese, pe baza unor cereri - precizarea operatorilor & și *, utilizați în C++ pentru operarea cu adrese de memorie, în contextul accesării valorii unei variabile atât prin numele său, cât și prin intermediul adresei acesteia - identificarea instrumentelor clasei list din Python pentru reprezentarea în memorie a unei liste înlănțuite, în contextul creării unei liste actualizate frecvent prin inserări și eliminări - recunoașterea instrumentelor limbajului de programare C++ care pot fi utilizate pentru reprezentarea listelor înlănțuite cu alocare statică, respectiv dinamică a memoriei pentru a evidenția flexibilitatea gestionării memoriei prin exemplul unei liste cu toți clienții unei firme de telefonie mobilă, cu numeroase actualizări - identificarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat sau orientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare - identificarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu liste de adiacență, respectiv a unui graf orientat, cu liste de succesori, în contextul parcurgerii nodurilor acestuia - identificarea instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore binar, astfel încât să permită accesul la fiii oricărui nod - identificarea instrumentelor limbajului de programare C++ pentru reprezentarea în memorie a nodurilor unui arbore binar, cu alocare dinamică, astfel încât să permită accesul la fiii oricărui nod	
CS 1.5. Identifică elementele de sintaxă și componentele din definiția claselor, pentru a le utiliza în implementarea algoritmilor în limbaj de programare cu programare orientată pe obiecte	
- identificarea convențiilor de numire a membrilor unei clase (public, privat) și sintaxa prin care sunt stabilite acestea, pentru a recunoaște controlul vizibilității, prin exemplul definirii unei clase „Rețea” unde atributele tehnice sunt private, iar metodele de întreținere sunt publice, pentru a reflecta controlul accesului la date sensibile - identificarea unor elemente de sintaxă pentru definirea unei clase, de exemplu, pentru clasa „Vehicul” cu atribute precum tip, viteză, stoc combustibil și metode pentru actualizarea vitezei sau realimentare - identificarea sintaxei de definire a constructorului unei clase, de exemplu, pentru inițializarea datelor unui utilizator pe o platformă de cursuri online, precum și a drepturilor pe care le are pe aceasta - identificarea sintaxei de definire a unei clase derivate, de exemplu, pentru particularizarea clasei „Vehicul” în clasa derivată „Camion”, cu atribute adăugate ca sarcină curentă și capacitate de transport, precum și metode pentru actualizarea sarcinii curente	

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 XI-a	
CS 2.1. Explică principiile de organizare a datelor în cadrul unor modele conceptuale complexe - liniare, rețea, ierarhice, obiectuale, pentru a le gestiona eficient	
- explicarea unor operații cu liste simplu înlănțuite (adăugare, eliminare a unui nod) pe baza unui exemplu de gestionare a listei de echipamente dintr-un club de robotică, grupate pe categorii, unde se pot adăuga deseori componente noi sau elimina echipamente nefuncționale	

Clasa a XI-a

- explicarea diferenței dintre listele simplu înălțuite, respectiv listele dublu înălțuite, pentru a înțelege avantajele parcurgerii bidirecționale, în cazul unei liste în care informația este memorată ordonat, ale cărei noduri se parcurg astfel încât aceasta să se obțină atât în ordine crescătoare, cât și în ordine descrescătoare
- explicarea particularităților metodelor de reprezentare a grafurilor (matrice de adiacență, liste de adiacență), în contextul reprezentării conexiunilor dintre localitățile identificate pe o hartă rutieră, respectiv a unei rețele logistice de alimentare cu apă a unei localități care conectează depozitele de apă cu dispozitivele de pompare a apei în rețea
- explicarea rolului elementelor componente ale unei diagrame prin care se reprezintă un graf orientat, respectiv un graf neorientat prin exemplul relațiilor dintre membrii unui grup de persoane: în cazul în care doar unii sunt populari, cunoscuți și de alți membri, chiar dacă ei nu îi cunosc pe aceștia (graf orientat) respectiv în cazul unor relații de prietenie reciprocă (graf neorientat)
- explicarea proprietăților unui graf bipartit, prin exemplul unei rețele de cooperare între unități de aprovizionare și centre logistice, unde fiecare conexiune leagă un nod din prima categorie (unități de aprovizionare) de un nod din a doua (centre logistice)
- explicarea relațiilor de descendență și ascendență într-un arbore cu rădăcină, prin exemplificarea cu arbori genealogici, cu limitare la linia paternă, în care ordinea descendenților nu este evidențiată, unde părinții și bunicii reprezintă relațiile de ascendență, iar copiii și nepoții reprezintă relațiile de descendență
- explicarea relațiilor ierarhice într-un arbore cu rădăcină, pe baza exemplului unei organizații în care fiecare angajat, cu excepția celor cu funcții de execuție, are colegi care îi sunt direct subordonați
- explicarea noțiunii de cheie și a condiției de ordine într-un arbore binar de căutare, prin exemplul utilizării unui criteriu de identificare (precum un cod unic) pentru indexarea și căutarea informațiilor într-un sistem de comunicații, unde fiecare cheie/cod are o poziție determinată în funcție de valoarea sa
- explicarea încapsulării, prin exemplul stabilirii membrilor unei clase, „Rețea”, unde se dorește ca datele tehnice să nu fie accesibile direct pentru utilizator, pentru a reflecta controlul asupra datelor sensibile
- explicarea moștenirii unei clase, punând în evidență „specializarea” acesteia, de exemplu pentru clasele „Autoturism” și „Camion” care sunt derivate din clasa „Vehicul”

CS 2.2. Explică etapele unor algoritmi specializați pe clase de probleme, pentru a fi putea utiliza în prelucrarea grafurilor, arborilor

- explicarea etapelor de funcționare ale algoritmului de parcurgere în lățime (BFS) a grafurilor, prin exemplul identificării rutei optime de comunicare între punctele de distribuție a unui produs (depozite) și punctele de desfacere (magazine), analizând ordinea vizitării nodurilor (depozite și magazine) și a legăturilor dintre acestea
- explicarea etapelor de funcționare ale algoritmului lui Dijkstra de determinare a unui drum de cost minim într-un graf, prin exemplul identificării rutei optime de transport între două puncte de distribuție a unui produs (depozite), având în vedere alegerea nodurilor (depozitelor) și actualizarea adecvată a distanțelor rutiere corespunzătoare legăturilor dintre acestea
- explicarea etapelor algoritmului lui Prim, prin exemplul construirii unei rețele de comunicații cu un cost total minim, integrând, după caz, pas cu pas, conexiunile cu costul cel mai mic
- explicarea modului în care algoritmul Roy-Floyd actualizează datele pentru determinarea drumurilor minime între oricare două noduri, prin exemplul simulării unei rețele logistice de transport, unde algoritmul determină rutele optime de aprovizionare între oricare două puncte de vânzare
- explicarea metodelor de parcurgere a arborilor binari, prin exemplul ilustrării modului în care datele despre organizarea unei excursii sunt procesate în ordine diferită, în funcție de scop: planificarea etapelor principale și a detaliilor (preordine) sau evaluarea și centralizarea costurilor finale (postordine)
- explicarea modului în care inserarea într-un heap menține proprietatea specifică, prin cernere, prin exemplul adăugării unei noi solicitări într-un sistem de prioritzare, unde se rearanjează automat ordinea solicitărilor în funcție de importanță

CS 2.3. Explică etapele unor strategii de tip Backtracking, Programare dinamică pentru rezolvarea problemelor

- explicarea modului de generare a tuturor soluțiilor utilizând metoda Backtracking, în contextul procesului de planificare a rutelor unei companii de transport aerian, unde fiecare destinație reprezintă o decizie dependentă de destinația anterioară
- explicarea rolului validării pe parcursul determinării soluțiilor cu metoda Backtracking, în contextul verificării condițiilor (acoperire, vizibilitate, siguranță) de amplasare a unor senzori într-o rețea de comunicații, în toate modurile posibile
- explicarea procesului de revenire la starea anterioară pentru generarea unei soluții cu metoda Backtracking, prin exemplul epuizării tuturor variantelor de alegere a unui dispozitiv adecvat pentru supravegherea unei anumite zone și necesitatea de a le reconfigura pe cele deja plasate
- explicarea modului de generare sistematică a tuturor soluțiilor utilizând metoda Backtracking generalizat, în contextul determinării tuturor traseelor printr-un labirint, în care zidurile sunt reprezentate prin valori 1, iar culoarele prin valori 0

Clasa a XI-a	
- explicarea rolului păstrării unor date relevante obținute pe parcursul aplicării metodei Programării dinamice, prin exemplul determinării celui mai lung subșir crescător al unui șir dat, determinând valori curente pe baza unor date relevante păstrate, obținute pe parcurs, în locul generării tuturor soluțiilor posibile și alegerea celei optime la final - explicarea principiului metodei Programării dinamice, de descompunere a problemei în subprobleme care se pot suprapune, prin exemplul determinării unei modalități de a plăti o sumă dată cu un număr minim de monede - explicarea principiului metodei Programării dinamice, de combinare a rezultatelor parțiale determinate, în contextul alegerii unor obiecte întregi (care nu pot fi tăiate), având valori și volume cunoscute, care să fie stocate într-un depozit cu o capacitate dată, astfel încât valoarea totală a celor stocate să fie maximă	
CS 2.4. Explică rolul principalelor elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele complexe - liniare, rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice, cu alocare statică și dinamică a memoriei	
- explicarea rolului variabilelor de tip pointer, adreselor de memorie și a alocării dinamice în C++, în contextul unei analogii cu procesul de alocare a unor locuințe de serviciu, aflate la diferite adrese, pe baza unor cereri - explicarea efectului operatorilor & și *, utilizați în C++ pentru operarea cu adrese de memorie, în contextul accesării valorii unei variabile atât prin numele său, cât și prin intermediul adresei acesteia - explicarea rolului instrumentelor clasei list pentru reprezentarea în memorie a unei liste înlănțuite, în contextul creării unei liste de participanți la un eveniment, actualizate frecvent prin inserări și eliminări - explicarea rolului instrumentelor limbajului de programare C++ care pot fi utilizate pentru reprezentarea listelor înlănțuite cu alocare statică, respectiv dinamică a memoriei pentru a evidenția flexibilitatea gestionării memoriei prin exemplul unei liste de împrumuturi a unei biblioteci, cu numeroase actualizări - explicarea rolului instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor dintr-un grup de persoane, în care sunt evidențiate relațiile de prietenie - explicarea modului în care pot fi utilizate instrumentele clasei list pentru reprezentarea în memorie a unui graf orientat, cu liste de predecesori, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de transmitere a unor informații, cu precizarea sursei - explicarea rolului instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore binar, astfel încât să permită accesul la fiii oricărui nod, în contextul unui arbore corespunzător unui chestionar, în care fiecare întrebare reprezintă un nod și este urmată, eventual, de o altă întrebare, în funcție de răspunsul dat (corect, incorect) - explicarea modului de acces la nodurile unui arbore binar cu instrumente ale limbajului de programare C++ în contextul în care acesta este memorat cu alocare dinamică, astfel încât să permită accesul la fiii oricărui nod	
CS 2.5. Explică rolul restricțiilor de acces și al componentelor unei clase, pentru a le utiliza în proiectarea claselor și implementarea algoritmilor cu programare orientată pe obiecte	
- explicarea modului în care constructorul inițializează starea unui obiect, pentru a crea corect instanțele, în contextul unui obiect „Concurs” (locație, disciplină, echipă) la crearea acestuia - explicarea modului de utilizare a membrilor clasei pentru a înțelege manipularea obiectelor într-un program, prin exemplul utilizării metodelor din clasa „Rețea” pentru a gestiona dispozitivele din rețea - explicarea principiului de moștenire a unei clase pentru reutilizarea și extinderea codului, prin exemplul clasei „Angajat”, extinsă prin clase derivate precum „Inginer” și „Profesor”, care moștenesc membrii comuni, dar adaugă membri specifici - explicarea relației de moștenire între clase pentru a interpreta reutilizarea și extinderea codului prin exemplul unei clase pentru autovehicule care transportă persoane și clasele derivate pentru autoturisme, respectiv pentru autocare	

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

Clasa a XI-a	
CS 3.1. Utilizează modul de organizare și prelucrare a datelor în cadrul unor modele conceptuale complexe - liniare, rețea, ierarhice, obiectuale, pentru a rezolva probleme de gestionare a datelor	
- aplicarea operațiilor de inserare și ștergere în diferite poziții în cadrul unei liste simplu înlănțuite, cu exemplificarea grafică a realizării legăturilor între nodurile conectate și evidențierea necesității prezenței acestora, prin exemplul realizării unui proiect școlar care adaugă și elimină activități - utilizarea unei liste dublu înlănțuite care să permită parcurgerea înainte și înapoi, în contextul dezvoltării unei aplicații care permite verificarea bidirecțională a etapelor dintr-un proiect școlar - utilizarea matricei de adiacență pentru a reprezenta un graf neorientat dat prin diagramă, în contextul modelării unei rețele de calculatoare conectate într-un laborator de informatică, evidențiind prezența valorilor nule pe diagonala principală, simetria față de aceasta - utilizarea listelor de succesori pentru a reprezenta un graf orientat dat prin matrice de adiacență, în contextul modelării fluxului de transmitere a datelor într-o rețea de comunicații	

Clasa a XI-a	
- utilizarea matricei de adiacență pentru a reprezenta un graf orientat dat prin diagramă în contextul modelării căilor de deplasare a autovehiculelor pe străzile unui oraș, având în vedere că acestea pot fi cu sens unic și că unesc intersecții, evidențiind prezența valorilor nule pe diagonala principală, asimetria față de aceasta - aplicarea metodei de reprezentare cu referințe descendente pentru un arbore dat prin diagramă, în contextul construirii unui arbore cu rădăcină pentru a reprezenta o structură organizatorică simplificată, cu niveluri corespunzătoare pentru diferite funcții - aplicarea metodei de reprezentare prin diagramă pentru un arbore dat prin referințe ascendente, în contextul determinării numărului de niveluri ale unui arbore cu rădăcină ce reprezintă o structură organizatorică simplificată, cu niveluri corespunzătoare pentru diferite funcții - aplicarea metodei de reprezentare cu referințe descendente (fii) pentru un arbore binar dat prin diagramă, în contextul determinării frunzelor acestuia - aplicarea încapsulării, prin exemplul stabilirii membrilor unei clase, „Rețea”, unde datele tehnice și metodele de întreținere caracterizează o rețea de calculatoare - aplicarea moștenirii, punând în evidență „specializarea” unei clase, de exemplu pentru clasele „Autoturism” și „Camion” care sunt derivate din clasa „Vehicul”	
CS 3.2. Aplică algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea grafurilor, arborilor	
- aplicarea algoritmilor BFS și DFS, pentru parcurgerea grafurilor neorientate, pe baza unei reprezentări prin diagramă a unui exemplu de graf neorientat, în care se marchează nodurile vizitate și se evidențiază particularitățile celor doi algoritmi - aplicarea algoritmului lui Dijkstra pentru a determina distanța minimă între două vârfuri, într-un graf orientat dat, prin exemplul identificării traseului optim între două orașe ținând cont de costul deplasării - aplicarea algoritmului lui Prim pentru a determina arborele de acoperire de cost minim într-un graf neorientat ponderat dat, prin exemplul proiectării unei rețele de transport care conectează toate sediile unei companii, cu un cost total minim de transport - aplicarea algoritmului Roy–Warshall pentru a determina matricea drumurilor într-un graf orientat, evidențiind obținerea soluției pentru oricare două vârfuri, în contextul rezolvării unei probleme de natură informatică - aplicarea algoritmului Roy–Floyd pentru a determina costurile minime ale drumurilor dintr-un graf orientat, prin exemplul calculării rutelor celor mai scurte de deplasare între punctele de vizitare dintr-un tur turistic tematic - aplicarea algoritmului lui Kruskal pentru a determina arborele de acoperire de cost minim într-un graf neorientat ponderat dat, prin exemplul îmbunătățirii unor linii bidirecționale de tramvai, dintre cele care conectează direct cartiere ale unui oraș, astfel încât cele alese pentru electrificare să asigure, direct sau indirect, conectarea oricăror două cartiere, cu un cost total minim - aplicarea metodelor de parcurgere (înordine, preordine, postordine) a nodurilor unui arbore binar, dat printr-o diagramă, evidențiind particularitățile fiecăreia și poziția, în șirul obținut, a rădăcinii arborelui/subarborelui pentru fiecare caz - aplicarea algoritmului de căutare a unei valori într-un arbore de căutare, prin exemplul căutării unui cuvânt într-un grup de cuvinte organizate sub forma unui arbore binar de căutare, evidențiindu-se particularitățile acestuia și numărul de comparații necesare - aplicarea algoritmilor pentru construirea și menținerea unui heap, pas cu pas, pentru un exemplu dat care simulează procesul de prioritizare a solicitărilor de intervenție ale echipelor de pompieri dintr-un oraș, unde fiecare inserare sau extragere din heap reflectă actualizarea nivelului de urgență al misiunilor	
CS 3.3. Aplică strategii de tip Backtracking, Programare dinamică pentru rezolvarea problemelor	
- aplicarea etapelor metodei Backtracking pentru generarea tuturor permutărilor unei mulțimi de patru cifre date, într-un context cotidian, prin exemplul generării tuturor codurilor posibile de câte 4 cifre distincte, pentru deblocarea unui dispozitiv, știind că utilizatorul își amintește că ultimele două cifre sunt pare - aplicarea etapelor metodei Backtracking pentru generarea tuturor combinațiilor unui set de obiecte într-un context cotidian, prin exemplul determinării tuturor posibilităților de a forma grupe de câte un număr fixat de echipamente necesare pentru un concurs de robotică - aplicarea etapelor metodei Backtracking pentru generarea sistematică a unor secvențe de valori, determinând eficient ultimele două soluții generate, prin exemplul planificării diferitelor scenarii de deplasare a unor drone care să includă toate punctele de interes - aplicarea etapelor metodei Programării dinamice pentru problema discretă a rucsacului într-un context de planificare a resurselor, prin exemplul selectării unor cadouri dintr-o listă dată, în funcție de greutatea și valoarea fiecăruia, astfel încât pachetul să aibă o greutate limitată, dată, și o valoare totală maximă - aplicarea metodei Programării dinamice pentru determinarea celui mai lung subșir comun a două șiruri de valori date, evidențiind utilizarea unor date memorate, calculate anterior, în cadrul unor subprobleme similare	

Clasa a XI-a	
CS 3.4. Utilizează principalele elemente ale limbajului de programare pentru prelucrarea datelor organizate în modele complexe - liniare, rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice, cu alocare statică și dinamică a memoriei	
- utilizarea instrumentelor clasei list pentru reprezentarea în memorie a unei liste înlănțuite, în contextul creării unei liste de melodii favorite, actualizate frecvent prin inserări și eliminări - utilizarea pointerilor în implementarea unui program C++ care declară o variabilă întreagă, un pointer către acea variabilă, și apoi afișează valoarea variabilei folosind pointerul - utilizarea pointerilor în implementarea unei funcții C++ care primește ca argument un pointer la o variabilă întreagă și modifică valoarea variabilei prin intermediul pointerului - utilizarea alocării dinamice a memoriei în C++ pentru a implementa o listă circulară, în contextul simulării rotației participanților într-un joc online - utilizarea alocării dinamice a memoriei în C++ pentru a implementa o listă dublu înlănțuită, în contextul parcurgerii acesteia pornind de la primul nod, respectiv de la ultimul nod - utilizarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de pe o hartă, în care sunt evidențiate relațiile de vecinătate dintre țări - utilizarea instrumentelor clasei list pentru reprezentarea în memorie a unui graf orientat, cu liste de succesori, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de transmitere a unor informații, cu precizarea destinației - utilizarea instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la tatăl oricărui nod, în contextul unui arbore asociat unui sistem de fișiere în care rădăcina reprezintă directorul principal, iar fiecare folder (sau fișier) are un singur director părinte - utilizarea alocării dinamice a memoriei în C++ pentru reprezentarea arborilor binari, în contextul implementării algoritmilor pentru construirea și parcurgerea unui astfel de arbore	
CS 3.5. Utilizează elementele de sintaxă și componentele din definiția claselor, în implementarea algoritmilor în limbaj de programare cu programare orientată pe obiecte	
- utilizarea elementelor de sintaxă ale limbajului de programare pentru definirea unei clase cu atribute și metode deja stabilite, pentru gestionarea unei situații reale, prin exemplul proiectării unei clase „ActivitateScolara” care gestionează datele despre activitățile dintr-un proiect școlar, de exemplu, codul activității, denumirea, obiectivul principal, numărul participanților etc. - utilizarea elementelor de sintaxă pentru crearea unei clase cu un constructor și metode pentru gestionarea unei situații reale, prin exemplul implementării unei clase „Autoturism” care include un constructor pentru inițializare (marcă, model, an de fabricație) și metode pentru îmbunătățirea performanțelor (de exemplu, tuning pentru motor), modificare a culorii - utilizarea elementelor de sintaxă pentru crearea unei clase care utilizează date ale unor clase definite anterior, de exemplu, clasa „Rezolvare” are drept membri obiecte din clasele „Elev”, respectiv „Problema”, asigurând corelarea dintre problemele propuse și elevii care le-au rezolvat - utilizarea elementelor de sintaxă pentru a defini clase derivate prin exemplul obținerii, din clasa de bază „JocVideo”, a claselor derivate „JocEducational” și „JocStrategie”, în scopul reutilizării componentelor comune și al extinderii acestora cu atribute și metode specifice fiecărui tip de joc	

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 XI-a	
CS 4.1. Analizează caracteristicile, modul de prelucrare, avantajele și dezavantajele utilizării unor modele conceptuale complexe - liniare, rețea, ierarhice, obiectuale, pentru a alege structura adecvată în rezolvarea unor probleme concrete	
- analizarea diferențelor dintre listele simplu și dublu înlănțuite pentru a înțelege avantajele parcurgerii bidirecționale, prin exemplul comparării între evidența deplasării unui tren (parcursere înainte) și verificarea inversă a traseului la întoarcere (parcursere înapoi) - analizarea avantajelor operației de inserare a unui element nou într-o listă simplu înlănțuită, față de implementarea listei printr-un tablou, prin exemplul adăugării unui membru nou într-o echipă a unui proiect educațional, în funcție de poziția acestuia în echipă - analizarea complexității operației de ștergere a unui element dintr-o listă simplu înlănțuită, incluzând localizarea acestuia, prin exemplul eliminării unui dispozitiv din lista de echipamente dintr-un laborator - analizarea caracteristicilor unui graf neorientat, conex, în contextul examinării traseului mașinii unei echipe care întreține indicatoarele rutiere din toate intersecțiile unei localități, pentru a verifica dacă mașina le poate accesa pe toate - examinarea gradelor nodurilor unui graf orientat asociat unei rețele de străzi dintr-un oraș pentru a identifica intersecțiile conectate la un număr mare de străzi, în vederea fluidizării traficului din localitate	

Clasa a XI-a	
- analizarea reprezentării prin diagramă a unui graf orientat asociat unei rețele de comunicații, în contextul stabilirii direcțiilor de transmitere a semnalelor - analizarea definițiilor echivalente ale unui arbore, în contextul rezolvării unei probleme de natură informatică, în care se verifică organizarea datelor sub formă de arbore, pentru a o alege pe cea mai ușor de implementat - analizarea relațiilor ierarhice în arborii cu rădăcină (înălțime, noduri descendente, noduri ascendente), în contextul comparării mai multor arbori asociați sistemului de foldere și subfoldere din calculator - analizarea diferențelor dintre metodele de reprezentare a unui arbore binar, având în vedere relațiile părinte-copil și proprietățile arborelui - analizarea unei structuri pentru a determina dacă are caracteristici specifice arborelui binar de căutare asociat organizării datelor elevilor unei clase, pentru a asigura accesul rapid și ordonat la informații, în funcție de numerele matricole ale elevilor - analizarea membrilor unei clase pentru a verifica dacă aceasta conține toate datele și metodele necesare pentru a răspunde unei funcționalități, prin exemplul clasei „Rețea”, care cuprinde datele tehnice și metodele de întreținere a unei rețele de calculatoare - analizarea oportunității de moștenire, punând în evidență „specializarea” unei clase, de exemplu pentru clasele „Autoturism” și „Camion” care sunt derivate din clasa „Vehicul”	
CS 4.2. Analizează comportamentul, aplicabilitatea, avantajele și dezavantajele utilizării unor algoritmi specializați pe clase de probleme pentru prelucrarea grafurilor, arborilor	
- analizarea metodelor BFS și DFS de parcurgere a grafurilor în vederea utilizării lor pentru verificarea proprietății de conexitate a unui graf neorientat, prin exemplul identificării componentelor izolate dintr-o rețea de comunicații bidirecționale, pentru a determina întreruperi în lanțul de transmitere a datelor - analizarea metodelor BFS versus DFS de parcurgere a unui graf neorientat/orientat în vederea adaptării lor pentru determinarea lanțului/drumului cu lungime minimă între două noduri ale grafului, respectiv pentru determinarea unui drum/lanț între două noduri ale grafului - analizarea rezultatelor parcurgerii în lățime a nodurilor unui graf neorientat, cu interpretarea ordinii nodurilor afișate, prin exemplul comparării ordinii de procesare a informațiilor într-un sistem de comunicații - analizarea complexității algoritmului Dijkstra, comparativ cu Roy-Floyd, prin exemplul evaluării performanței sistemului de planificare a rutelor unui traseu turistic tematic, în care se cunosc distanțele dintre obiective, în funcție de numărul de puncte de destinație și de conexiuni disponibile - analizarea complexității algoritmului Roy-Warshall, în contextul determinării matricei drumurilor corespunzătoare unei rețele de șosele între diverse localități aflate pe un traseu de transport - analizarea rezultatelor algoritmului lui Prim, respectiv Kruskal, în contextul determinării acoperirii de cost minim a unui graf ponderat, prin exemplul construirii unui model grafic care indică legăturile logistice de bază necesare pentru conectarea tuturor sediilor unei firme de transport - compararea metodelor de parcurgere a arborilor binari pentru a determina situațiile optime de utilizare a acestora prin exemplul alegerii metodei de parcurgere adecvate într-un sistem de planificare a activităților - analizarea modului în care inserarea unui nod afectează structura generală a unui arbore binar de căutare prin exemplul inserării unor elevi care s-au alăturat colectivului clasei - analizarea efectului schimbării ordinii de inserare a cheilor într-un arbore binar de căutare, prin exemplul examinării modului în care această ordine afectează structura arborelui - analizarea transformărilor structurii unui heap, prin vizualizarea pas cu pas a algoritmului de formare a unui ansamblu, prin exemplul simulării modului în care se reordonează automat prioritățile sarcinilor pe măsură ce se adaugă sarcini noi, cu o prioritate stabilită - compararea performanței metodei de sortare cu heap (Heapsort), cu alte metode de sortare studiate anterior, prin exemplul comparării vitezei de procesare în sortarea datelor	
CS 4.3. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de tip Backtracking, Programare dinamică pentru rezolvarea problemelor	
- analizarea mecanismelor de avans și de revenire în arborele soluțiilor la aplicarea metodei Backtracking, prin exemplul generării tuturor anagramelor cu anumite proprietăți ale unui cuvânt format cu litere distincte (de exemplu, astfel încât literele să nu fie alăturate și în cuvântul dat), pentru evidențierea modului în care algoritmul explorează adecvat posibilitățile și revine pentru a face alte alegeri de litere, în vederea obținerii de noi soluții - analizarea avantajelor și dezavantajelor aplicării metodei Backtracking comparativ cu alte metode de generare pentru a evalua eficiența acestora în obținerea tuturor numerelor naturale cu anumite proprietăți de aranjare a cifrelor, în funcție de numărul acestora - analizarea aplicabilității metodei Backtracking, în funcție de numărul de soluții posibile, prin exemplul evaluării timpului de executare pentru planificarea tuturor rutelor aeriene pentru mai multe avioane care zboară simultan	

Clasa a XI-a	
- analiza aplicabilității metodei Backtracking în plan pentru obținerea tuturor traseelor posibile care pot fi urmate pentru a părăsi un labirint, în care zidurile și culoarele sunt marcate adecvat în cadrul unei matrice - analiza complexității unei soluții care utilizează metoda Programării dinamice în raport cu o soluție generată cu metoda Backtracking și determinarea optimului, pentru a determina cel mai lung subșir crescător al unui șir dat - analiza aplicabilității metodei Programării dinamice (care oferă rezultatul corect și eficient, în raport cu metoda Backtracking cu determinarea pe parcurs a optimului, care oferă rezultatul corect, dar lent, respectiv în raport cu metoda Greedy, care nu garantează un rezultat corect), pentru a determina obiectele întregi care se pot transporta cu un rucsac de o anumită capacitate, astfel încât valoarea lor totală să fie maximă	
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 complexe - liniare, rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice	
- analiza avantajelor și dezavantajelor utilizării instrumentelor clasei list pentru reprezentarea în memorie a unei liste înlănțuite, în contextul creării unei liste de jocuri video favorite, actualizate frecvent prin inserări și eliminări - analiza aplicabilității pointerilor în C++ în contextul examinării unui fragment de cod pentru identificarea potențialelor erori - analiza aplicabilității pointerilor în implementarea unei funcții C++ care primește ca argument un pointer la o variabilă întreagă și modifică valoarea variabilei prin intermediul pointerului - analiza avantajelor și dezavantajelor utilizării alocării dinamice a memoriei în C++ pentru a implementa o listă circulară, în contextul simulării rotației planetelor într-un sistem planetar - analiza avantajelor și dezavantajelor alocării dinamice a memoriei în C++ pentru a implementa o listă dublu înlănțuită, în contextul parcurgerii acesteia pornind de la primul nod, respectiv de la ultimul nod - analiza instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență sau cu liste de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de identificare a calculatoarelor dintr-o rețea, în care sunt evidențiate relațiile de conexiune directă dintre acestea - analiza instrumentelor clasei list pentru reprezentarea în memorie, cu liste de predecesori, a unui graf orientat corespunzător unui sistem de transmitere a unor informații, în care vârfurile reprezintă stații de emisie-recepție și pentru fiecare se precizează sursele de la care primește direct informații, în contextul determinării, pentru o stație dată, a tuturor surselor directe sau indirecte de informații - analiza instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la tatăl oricărui nod, în contextul unui arbore asociat unui sistem ierarhic din cadrul unei organizații în care rădăcina corespunde directorului general, fiecare alt angajat are un singur manager direct, iar în organizație nu există cicluri de subordonare - analiza aplicabilității alocării dinamice a memoriei în C++ pentru reprezentarea arborilor binari, în contextul implementării algoritmilor pentru construirea și parcurgerea unui astfel de arbore	
CS 4.5. Analizează aplicabilitatea, avantajele și dezavantajele utilizării programării orientate pe obiecte, în implementarea algoritmilor în limbaj de programare	
- analiza avantajelor și dezavantajelor utilizării convențiilor de numire a membrilor unei clase pentru a interpreta impactul asupra securității datelor, prin exemplul protejării datelor sensibile într-o aplicație de evidență a angajaților dintr-o instituție (date de identificare, adresă de domiciliu) - analiza unei aplicații cu mai multe clase pentru a identifica dependențele și interacțiunile dintre obiecte, prin exemplul analizării relațiilor dintre clasele „Carte”, „Cititor”, „Biblioteca”, pentru o bibliotecă publică - analiza relațiilor de moștenire pentru a detecta eventuale probleme de redundanță în cod, prin exemplul verificării claselor „Vehicul”, „Camion”, „Autocar”, pentru a evita duplicarea atributelor comune (de exemplu, viteză, autonomie, echipaj) - analiza unei aplicații care conține clase și clase derivate pentru a identifica definițiile metodelor apelate prin diferite obiecte, prin exemplul unei aplicații care gestionează persoanele dintr-o școală și care conține clase ca „Persoana”, „Elev”, „Profesor”	

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

Clasa a XI-a
CS 5.1. Argumentează alegerea unor modele conceptuale complexe - liniare, rețea, ierarhice, obiectuale și a modurilor de prelucrare a acestora pentru rezolvarea unor probleme informatice concrete

Clasa a XI-a	
- argumentarea utilizării unei liste simplu înlănțuite în contextul descrierii pașilor de rezolvare algoritmică a unei probleme, astfel încât fiecare pas să descrie o operație care trebuie să urmeze în mod precis după o altă operație - argumentarea utilizării unei liste circulare simplu înlănțuite, în contextul aranjării unui grup de copii la o masă circulară, pentru a participa la un joc de tip ștafetă, și eliminarea din joc, pe rând, a câte unui copil, cu reorganizarea participanților rămași în joc - evaluarea eficienței operațiilor de inserare și de eliminare a unui element dintr-o listă simplu înlănțuită, comparativ cu o listă dublu înlănțuită, respectiv cu o listă secvențială, având în vedere timpul necesar pentru aceste operații - argumentarea alegerii metodelor de reprezentare a grafurilor neorientate prin matrice de adiacență sau liste de adiacență, în contextul alegerii modului optim de reprezentare a unei rețele de transport aerian, având în vedere numărul destinațiilor și densitatea conexiunilor - evaluarea alegerii modelării printr-un graf orientat a unei situații reale, prin exemplul validării unei diagrame de proces care descrie legăturile condiționale (dependențe) dintre etapele unei activități - argumentarea alegerii unei metode de reprezentare a arborilor cu rădăcină (referințe descendente, referințe ascendente), în funcție de scopul utilizării, prin exemplul alegerii metodei optime de reprezentare a structurii unei echipe care va participa la un concurs de robotică, în funcție de necesitatea de prezentare ierarhică a componenței echipei sau de detaliere a rolului fiecărui membru - argumentarea utilizării unui arbore binar de căutare în contextul căutării unor informații într-o mulțime dată, în mod repetat - argumentarea alegerii membrilor unei clase, astfel încât aceasta să conțină toate datele și metodele necesare pentru a răspunde unei funcționalități, prin exemplul clasei „Rețea”, care cuprinde datele tehnice și metodele de întreținere a unei rețele de calculatoare - argumentarea alegerii moștenirii, punând în evidență „specializarea” unei clase, de exemplu pentru clasele „Autoturism” și „Camion” care sunt derivate din clasa „Vehicul”	
CS 5.2. Evaluează corectitudinea și eficiența soluțiilor cu algoritmi specializați pe clase de probleme pentru prelucrarea grafurilor, arborilor	
- evaluarea menținerii proprietății de conexitate utilizând metodele de parcurgere a unui graf neorientat în contextul unei rețele de comunicații având în vedere posibilitatea menținerii legăturii între dispozitive în caz de pierdere a unui nod (nod critic), prin înlăturarea unui dispozitiv - evaluarea eficienței algoritmului lui Dijkstra pentru determinarea drumurilor de cost minim într-un graf ponderat, prin exemplul determinării rutelor optime de aprovizionare între un depozit și punctele de vânzare - evaluarea eficienței algoritmului lui Prim pentru determinarea acoperirii de cost minim într-un graf ponderat, prin exemplul determinării rețelei logistice de transport cu cost minim - evaluarea avantajelor și dezavantajelor alegerii unuia dintre algoritmii Roy-Floyd sau Dijkstra în funcție de structura grafului ponderat, prin exemplul alegerii algoritmului optim pentru o rețea de comunicații în funcție de numărul de conexiuni (numărul de noduri), densitatea (numărul de muchii) și costurile de comunicare - evaluarea corectitudinii alegerii uneia dintre metodele de parcurgere ale arborilor binari, în contextul specificului unui sistem ierarhic: preordine pentru diseminarea rapidă a sarcinilor, postordine pentru evaluarea completă, inordine pentru generarea rapoartelor cu elemente sortate - evaluarea complexității algoritmului de căutare într-un arbore binar de căutare comparativ cu căutarea liniară, prin exemplul comparării timpului mediu necesar pentru regăsirea unui cod într-o structură ordonată după chei versus o listă nesortată - evaluarea complexității algoritmului de sortare cu ansambluri (Heapsort), prin exemplul estimării timpului de sortare a unei liste mari de resurse (combustibil, echipamente), analizând eficiența metodei în condiții de volum ridicat de date	
CS 5.3. Evaluează corectitudinea și eficiența algoritmilor care utilizează strategii de tip Backtracking, Programare dinamică pentru rezolvarea problemelor	
- evaluarea corectitudinii aplicării metodei Backtracking pentru probleme de generare, prin exemplul determinării modului de parcurgere, într-o croazieră, a unor insule, astfel încât fiecare dintre acestea să fie vizitată o singură dată (cu excepția celei de plecare, la care se și încheie excursia), iar vasul să urmeze doar trasee acceptate între două insule vizitate consecutiv și să aducă turiștii, la final, în punctul de plecare - evaluarea corectitudinii aplicării metodei Backtracking pentru generarea tuturor rutelor de livrare, cu anumite restricții privind ordinea punctelor parcurse pe traseu, prin exemplul comparării eficienței explorării exhaustive a variantelor de traseu și verificarea finală a restricțiilor impuse, comparativ cu eliminarea pe parcurs a unor trasee necorespunzătoare - evaluarea complexității unui algoritm care folosește metoda Backtracking, prin exemplul estimării timpului necesar pentru explorarea tuturor variantelor de poziționare a unor echipamente video într-o rețea de comunicații - evaluarea corectitudinii aplicării metodei Backtracking în plan pentru generarea sistematică a soluțiilor, prin exemplul rezolvării unui puzzle Sudoku	

Clasa a XI-a	
- evaluarea aplicabilității metodei Programării dinamice pentru probleme cu subprobleme suprapuse, prin exemplul analizării modului în care un robot poate traversa un teren cu un consum total minim de energie, mergând doar spre dreapta sau în jos și consumând o anumită cantitate de energie pentru fiecare zonă vizitată - evaluarea corectitudinii și eficienței algoritmilor care utilizează metoda Programării dinamice în contextul determinării numărului de moduri în care poate fi urcată o scară, cu sărituri peste câte una sau câte două trepte, comparând abordarea recursivă cu cea bazată pe memorarea rezultatelor parțiale, pentru a identifica avantajele reutilizării soluțiilor parțiale - evaluarea eficienței utilizării metodei Programării dinamice pentru probleme cu subprobleme suprapuse, în contextul reducerii timpului de executare prin reutilizarea soluțiilor parțiale într-o aplicație de selectare a celui mai lung subșir crescător al unui șir	
CS 5.4. Evaluează corectitudinea și eficiența aplicațiilor care utilizează elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele complexe - liniare, rețea, ierarhice, în vederea rezolvării eficiente a problemelor informatice	
- evaluarea utilizării pointerilor în C++ pentru o anumită sarcină de programare, luând în considerare factori precum eficiența, complexitatea codului și riscul de erori - evaluarea corectitudinii utilizării pointerilor în implementarea în C++ a listelor înlănțuite cu alocare dinamică a memoriei în contextul unei liste de contacte, identificând și prevenind erorile frecvente (pierderi de memorie, accesări invalide) - evaluarea corectitudinii și eficienței implementării structurilor de date liniare înlănțuite, având în vedere compararea listelor simplu și dublu înlănțuite din perspectiva consumului de memorie și a vitezei operațiilor de inserare/ștergere într-o aplicație de gestiune a evidenței elevilor, unde numărul datelor memorate variază frecvent - evaluarea aplicabilității instrumentelor clasei list pentru reprezentarea în memorie a unui graf neorientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de identificare a unor persoane dintr-un grup, evidențiind relațiile de prietenie dintre acestea - evaluarea instrumentelor clasei list pentru a le alege pe cele adecvate pentru reprezentarea în memorie a unui graf orientat, cu liste de succesori, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de transmitere a unor informații - argumentarea utilizării instrumentelor clasei dict pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la descendenții direcți ai oricărui nod, în contextul unui arbore asociat unui sistem ierarhic din cadrul unei organizații în care rădăcina corespunde directorului general, unii angajați au mai mulți subordonați direcți, iar în organizație nu există cicluri de subordonare - evaluarea aplicabilității alocării dinamice a memoriei pentru reprezentarea în C++ a arborilor binari, în contextul implementării algoritmilor pentru construirea și parcurgerea unui astfel de arbore - evaluarea eficienței, din punctul de vedere al gestionării memoriei, a implementării în C++ a unui arbore binar de căutare cu alocare dinamică a memoriei, comparativ cu alocarea statică a memoriei	
CS 5.5. Evaluează corectitudinea și eficiența programelor care utilizează programare orientată pe obiecte, pentru a rezolva probleme informatice	
- evaluarea corectitudinii definirii și utilizării claselor pentru organizarea datelor în contextul implementării unei aplicații de evidență a echipamentelor IT dintr-o școală, comparând varianta în care au fost definite clase față de organizarea cu funcții și variabile globale, din perspectiva clarității și modularității codului, cu avantaje pentru întreținerea acestuia și a prevenirii erorilor la actualizarea datelor - evaluarea corectitudinii aplicării convențiilor de numire pentru protecția informațiilor sensibile într-o aplicație reală, prin exemplul unei clase pentru gestionarea datelor personale ale elevilor, comparând accesul liber la date cu utilizarea unor date interne (protejate prin convenții) și a unor metode publice de acces, pentru validarea modificărilor și menținerea consistenței datelor - evaluarea corectitudinii definirii unei clase cu un constructor și metode pentru gestionarea unei situații reale, prin exemplul implementării unei clase „CursOnline” care include un constructor pentru inițializare (denumire, perioadă, număr de ore petrecute online, cost) și metode pentru actualizarea numărului de cursanți sau pentru modificarea perioadei - evaluarea corectitudinii definirii unei clase care are drept membri obiecte ale unor clase definite anterior, de exemplu, clasa „Optiune” are drept membri obiecte ale claselor „Client”, respectiv „Excursie”, asigurând corelarea dintre excursiile propuse și clienții care s-au înscris la acestea - evaluarea corectitudinii și eficienței utilizării moștenirii în contextul comparării implementării cu moștenire față de clase independente cu cod duplicat pentru gestionarea membrilor unei comunități școlare (elevi, profesori, personal), analizând reducerea redundanței codului, ușurința în modificare și extindere și menținerea consistenței datelor	

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

Clasa a XI-a**CS 6.1. Proiectează algoritmi personalizați care utilizează modele conceptuale complexe - liniare, rețea, ierarhice, obiectuale, pentru organizarea și prelucrarea eficientă a datelor, utilizând algoritmi de bază, în vederea rezolvării unor probleme**

- proiectarea unui algoritm care utilizează o listă liniară simplu înlănțuită pentru a gestiona pacienții unui medic, care să permită adăugarea datelor unui pacient la finalul listei, la înscrierea acestuia, eliminarea datelor unui anumit pacient și afișarea datelor pacienților, în ordinea înscrierii
- proiectarea unui algoritm care utilizează o listă circulară simplu înlănțuită pentru un sistem de planificare a sarcinilor repetitive, care gestionează participanții la un joc, pe runde, în care jucătorii acționează în ordinea dată, iar după ce ultimul jucător și-a încheiat acțiunea, jocul continuă automat, cu o nouă rundă, începând cu primul jucător
- proiectarea unui algoritm care utilizează grafuri neorientate pentru a gestiona datele referitoare la țările dintr-o regiune, determinând vecinii fiecăreia, țările cu cei mai mulți vecini sau țările izolate, înconjurate de ape internaționale în afara apelor teritoriale proprii
- proiectarea unui algoritm care utilizează grafuri orientate în contextul modelării căilor de deplasare a autovehiculelor pe străzile unui oraș, având în vedere că acestea pot fi cu sens unic și că unesc intersecții, determinând străzile cu sens unic și punctele reprezentând extremități ale unor străzi fără ieșire/fundături
- proiectarea unui algoritm pentru modelarea unui meniu arborescent, utilizând un arbore cu rădăcină în care fiecare nod corespunde opțiunii pentru o anumită acțiune, cu posibilitatea de a opta și pentru acțiuni subsecvente acesteia
- proiectarea unui algoritm care utilizează arbori ordonați, determinând înălțimea arborelui, nodurile frunză, în contextul modelării unui arbore genealogic al unei familii
- proiectarea unui algoritm care utilizează arborele binar de căutare pentru a identifica rapid datele dintr-un depozit de produse de papetărie pe baza codului unic de inventar
- proiectarea unui algoritm care utilizează un ansamblu (min-heap) pentru prioritizarea automată a cererilor de aprovizionare a unui depozit de alimente, ordonându-le în funcție de stocurile existente
- proiectarea unui algoritm pentru un model obiectual care încapsulează date și metode, prin exemplul determinării numărului total de exemplare ale publicațiilor dintr-o bibliotecă, în care o publicație este definită ca un model obiectual (titlu, autor, număr de exemplare și metode pentru modificarea numărului de exemplare), iar o bibliotecă este definită ca un model obiectual (listă de publicații, metode pentru adăugarea unei noi publicații)
- proiectarea unui algoritm pentru un model obiectual cu clase derivate, punând în evidență utilizarea unor date și metode moștenite, precum și a unor date și metode specifice claselor derivate, de exemplu pentru clasele „Autoturism” și „Camion” derivate din clasa „Vehicul”

CS 6.2. Proiectează soluții cu aplicarea personalizată a algoritmilor specializați pe clase de probleme pentru prelucrarea grafurilor, arborilor

- proiectarea unui algoritm care utilizează metoda de parcurgere în lățime a nodurilor unui graf prin exemplul unui centru de coordonare pentru situații de urgență care trebuie să trimită rapid echipe medicale către toate localitățile, pornind din orașul în care se află centrul, în situația unor intervenții
- proiectarea unui algoritm care utilizează metoda de parcurgere în adâncime a nodurilor unui graf în contextul determinării unui traseu de la un punct de acces către un punct dat, printr-o rețea de tuneluri sau galerii miniere ale unei mine vechi
- proiectarea unui algoritm pentru a determina dacă un graf este bipartit, utilizând metodele BFS sau DFS, prin exemplul împărțirii unei rețele de comunicații în două grupuri funcționale, respectiv emițătoare și receptoare, pentru o mai bună gestionare a traficului de date
- proiectarea unui algoritm pentru calcularea costurilor minime de livrare pentru o rețea de depozite, unități de desfacere și rute modelate cu un graf ponderat, utilizând algoritmul lui Dijkstra sau Roy Floyd, prin exemplul optimizării transportului de la un depozit regional și unitățile de desfacere, pe baza distanțelor și costurilor de deplasare
- proiectarea unui algoritm care utilizează algoritmul lui Prim prin exemplul proiectării unei rețele de iluminare stradală la care să fie conectați, prin cabluri, toți stâlpii de iluminat din oraș, iar lungimea totală a cablurilor utilizate să fie minimă
- proiectarea unui algoritm care, pornind de la două șiruri de valori, reprezentând nodurile aceluiași arbore binar, parcurse în preordine, respectiv inordine, determină nodurile în ordinea parcurgerii lor în postordine
- crearea unei aplicații pentru simularea căutării într-un arbore binar de căutare pentru identificarea unui elev într-o bază de date școlară, după criteriul mediei, iar la medii egale, alfabetic, după nume
- crearea unei aplicații care efectuează sortarea cu ansambluri (Heapsort), pentru ordonarea produselor dintr-un magazin în funcție de popularitate

Clasa a XI-a	
CS 6.3. Proiectează algoritmi cu aplicarea personalizată a strategiilor de tip Backtracking, Programare dinamică pentru rezolvarea problemelor	
- proiectarea unui algoritm care rezolvă o problemă folosind metoda Backtracking, pentru planificarea poziționării camerelor de supraveghere într-o clădire, având în vedere anumite restricții - proiectarea unui algoritm folosind metoda Backtracking pentru a determina toate modalitățile de a obține buchete de flori dintr-o mulțime dată, astfel încât acestea să aibă culori diferite - crearea unei aplicații care rezolvă o problemă folosind metoda Backtracking generalizată, pentru a determina toate traseele pe care le poate urma un robot într-un depozit, pentru a localiza un colet - proiectarea unui algoritm care utilizează metoda Programării dinamice pentru a calcula rapid numărul de combinații posibile pentru o parolă de un număr dat de caractere din mulțimea $\{a, b, 1, 2\}$, cu restricții simple de plasare a literelor aflate pe poziții consecutive, evitând recalculările inutile - proiectarea unui algoritm pentru planificarea bugetului folosind metoda Programării dinamice, pentru distribuirea fondurilor disponibile, maximizând eficiența beneficiilor obținute prin alegerile făcute, în limitele bugetului, prin identificarea subproblemelor suprapuse pentru a obține o soluție optimă fără recalculări inutile	
CS 6.4. Implementează aplicații care integrează în mod personalizat elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele complexe - liniare, rețea, ierarhice, în vederea realizării unor soluții funcționale și performante	
- implementarea listelor înlănțuite, având în vedere operații de inserare/ștergere într-o aplicație de gestiune a evidenței elevilor, unde numărul datelor memorate variază frecvent - implementarea unui algoritm pentru stocarea evenimentelor, în ordine cronologică, într-un sistem de jurnalizare utilizând liste liniare cu alocare dinamică a memoriei - implementarea unui algoritm în C++ pentru gestionarea matricelor rare folosind pointeri și alocare dinamică a memoriei pentru a optimiza utilizarea spațiului - implementarea unei aplicații care utilizează clasa list pentru reprezentarea în memorie a unui graf orientat, cu matrice de adiacență, în contextul creării structurii corespunzătoare pentru memorarea datelor de identificare a unor persoane dintr-un grup, evidențind modul în care o persoană cunoaște sau nu o altă persoană - implementarea unei aplicații care utilizează clasa list pentru reprezentarea în memorie a unui graf neorientat, cu liste de muchii, în contextul parcurgerii nodurilor unui graf corespunzător unui sistem de comunicare între prieteni - implementarea unei aplicații care utilizează instrumentele clasei list pentru reprezentarea în memorie a nodurilor unui arbore cu rădăcină, astfel încât să permită accesul la ascendentul direct al oricărui nod, în contextul unui arbore asociat unui sistem ierarhic din cadrul unei organizații în care rădăcina corespunde directorului general, iar fiecare alt angajat are un singur manager - elaborarea unei aplicații cu arbori binari de căutare cu alocare dinamică a memoriei pentru gestionarea rapidă a comenzilor unei firme de livrare de produse, identificate prin chei unice	
CS 6.5. Implementează aplicații în limbaj de programare care integrează clase personalizate, pentru a modulariza și organiza eficient codul în cadrul soluțiilor informatice	
- implementarea unei aplicații de gestionare a bibliotecii școlare, cu clase pentru cărți, utilizatori și evidența împrumuturilor - implementarea unei aplicații de evidență a comenzilor într-un magazin online, cu clase pentru produse și depozit - implementarea unui simulator de joc pentru participarea la concursul de robotică, având clase pentru „Echipament”, „Robot” și „Elev” - implementarea unei aplicații pentru transportul elevilor la școală, care utilizează clasele „Ruta”, „Masina”, „Autobuz” care să conțină metode și mecanismul de moștenire	

CONȚINUTURI ALE ÎNVĂȚĂRII

Clasa a XI-a

Domenii de conținut	Conținuturi
1. Organizarea conceptuală a datelor	1.1. Modelul conceptual liniar – listă - caracteristici ale unei liste înlănțuite, tipuri de liste înlănțuite (simplu înlănțuite, dublu înlănțuite, liste circulare simplu înlănțuite, liste circulare dublu înlănțuite); - operații de bază într-o listă înlănțuită: adăugare a unui nod, eliminare a unui nod; - repere pentru parcurgerea nodurilor unei liste înlănțuite și aplicarea algoritmilor de bază. 1.2. Modelul conceptual rețea - graf - concepte de bază și caracteristici: graf neorientat și orientat, nod și vârf, muchie și arc, adiacență, incidență, grad, grad interior, grad exterior, lanț și drum, lanț elementar și drum elementar, lanț simplu și drum simplu, lanț compus și drum compus, ciclu și circuit, ciclu elementar și circuit elementar; - metode de reprezentare a unui graf: diagramă, matrice de adiacență, liste de adiacență, liste de succesori, liste de predecesori, liste de muchii, liste de arce; - concepte de bază și caracteristici ale unor grafuri asociate unui graf: subgraf, graf parțial, graf transpus; - concepte de bază, caracteristici și proprietăți ale unor tipuri de grafuri: graf complet, graf bipartit, graf conex (puncte critice, muchii critice), graf tare conex (matricea drumurilor), graf ponderat (matricea costurilor), graf hamiltonian (lanț hamiltonian, ciclu hamiltonian), graf eulerian (lanț eulerian, ciclu eulerian); - repere pentru aplicarea algoritmilor de bază în prelucrarea grafurilor. 1.3. Model conceptual ierarhic - arbore - caracteristici și definiții echivalente ale unui arbore, respectiv arbore cu rădăcină; - concepte de bază într-un arbore cu rădăcină și caracteristici ale acestora: rădăcină, nivel al unui nod, nod frunză, nod descendent direct/fiu, nod descendent, nod ascendent direct/tată, nod ascendent, nod frate; - metode de reprezentare a unui arbore cu rădăcină: diagramă, referințe ascendente, referințe descendente; - caracteristici ale arborilor ordonați, arborilor binari (cu subtipul arbore binar de căutare, arbore binar complet, ansamblu - heap); - metode de reprezentare a unui arbore binar cu referințe descendente (liste de fii); - repere pentru aplicarea algoritmilor de bază pentru prelucrarea arborilor. 1.4. Model conceptual obiectual – clase și obiecte - concepte avansate și caracteristici: clasă, obiect, moștenire, clasă de bază, clasă derivată; - repere pentru aplicarea algoritmilor de bază cu date organizate obiectual.
2. Strategii de rezolvare a problemelor	2.1. Metoda Backtracking - caracteristici ale metodei Backtracking; - repere pentru aplicarea metodei de generare sistematică a unor secvențe de valori; - caracteristici ale metodei Backtracking generalizate; - repere pentru aplicarea metodei Backtracking generalizate pentru generarea sistematică a unor secvențe de perechi de valori. 2.2. Metoda Programării dinamice - caracteristici ale metodei Programării dinamice; - repere pentru aplicarea metodei. 2.3. Algoritmi specifici pentru prelucrarea grafurilor - caracteristici, etape ale metodelor de parcurgere în lățime (Breadth First Search - BFS), respectiv în adâncime (Depth First Search – DFS) și repere pentru aplicarea acestora; - repere pentru utilizarea metodelor de parcurgere pentru verificarea proprietății de conexitate a unui graf neorientat și determinarea componentelor conexe; - repere pentru utilizarea metodelor de parcurgere pentru determinarea matricei drumurilor/lanțurilor unui graf; - caracteristici, etape ale algoritmului Roy-Warshall pentru determinarea matricei drumurilor/lanțurilor unui graf și repere pentru aplicarea acestuia; - repere pentru utilizarea matricei drumurilor pentru verificarea proprietății de tare conexitate și determinarea componentelor tare conexe ale unui graf orientat;

Domenii de conținut	Conținuturi
	- caracteristici, etape ale algoritmului lui Dijkstra pentru determinarea drumurilor de cost minim într-un graf ponderat și repere pentru aplicarea acestuia; - caracteristici, etape ale algoritmului Roy-Floyd pentru determinarea drumurilor de cost minim într-un graf ponderat și repere pentru aplicarea acestuia; - repere pentru adaptarea metodei de parcurgere în adâncime pentru determinarea unui ciclu eulerian; - caracteristici, etape ale algoritmului lui Prim pentru determinarea arborelui de acoperire de cost minim într-un graf ponderat și repere pentru aplicarea acestuia; - caracteristici, etape ale algoritmului lui Kruskal pentru determinarea acoperirii de cost minim într-un graf ponderat și repere pentru aplicarea acestuia. 2.4. Algoritmi specifici pentru prelucrarea arborilor - caracteristici, etape ale metodelor de parcurgere a nodurilor unui arbore binar în preordine, inordine și postordine și repere pentru aplicarea acestora; - caracteristici, etape ale metodelor de căutare și de inserare a unei chei într-un arbore binar de căutare și repere pentru aplicarea acestora; - caracteristici, etape ale algoritmilor specifici pentru prelucrarea unui ansamblu (construire a ansamblului, inserare, respectiv eliminare a unui nod, sortare cu ansambluri) și repere pentru aplicarea acestora.
3. Memorarea datelor și organizarea codului în limbaj de programare	3.1. Programare orientată pe obiecte - sintaxa definirii unei clase, niveluri de acces la membrii clasei, constructor, destructor/finalizer, clasă derivată în Python; - sintaxa definirii unei clase, niveluri de acces la membrii clasei, constructor, destructor, supraîncărcare a metodelor, clasă derivată în C++; - repere pentru definirea unor clase simple și a claselor derivate. 3.2. Pointeri în C++ - caracteristici, rol, declarare a unor variabile de tip pointer, alocare și eliberare dinamică a memoriei; - operații și operatori specifici: adresă la care se memorează valoarea unei variabile, acces la conținutul memorat la o adresă, atribuire, adunare cu un număr întreg, scădere a două adrese; - repere pentru prelucrarea listelor înlănțuite și a arborilor binari cu alocare dinamică a memoriei. 3.3. Paradigme de bază în programare - caracteristici, tipuri de paradigme (de exemplu, procedurală, obiectuală, declarativă); - concepte, funcționalități, avantaje și dezavantaje ale altor limbaje de programare (de exemplu, C, Java, JavaScript, C#, J#).

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 măsoară? 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.

Un context de învățare poate viza simultan mai multe competențe, mai ales atunci când acestea sunt interconectate. De exemplu, pentru rezolvarea unei probleme cu ajutorul calculatorului se vizează, în cadrul aceleiași activități de învățare, atât competențe de nivel cognitiv scăzut (identificarea principalelor caracteristici ale modelelor conceptuale și operaționale implicate, explicarea principiilor care stau la baza acestora, în contextul particular al problemei, utilizarea lor în rezolvare) dar și competențe de nivel cognitiv ridicat (analizarea caracteristicilor modelelor conceptuale și operaționale specifice problemei, evaluarea corectitudinii și eficienței soluțiilor propuse și elaborarea algoritmului personalizat).

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 algoritmi 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 informațional — 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 algoritmi de bază (pentru numărare, sumă, produs, minim, maxim, prima sau ultima valoare cu o anumită proprietate, verificarea unor proprietăți) studiate 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 elementele 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).

Pentru specializarea matematică-informatică, la clasele cu predarea disciplinei informatică în regim intensiv, **limbajele de programare pentru implementare sunt Python și C++**, de exemplu în una dintre variantele:

- în paralel (alocând pentru practică anumite ore, săptămânal, fiecărui limbaj, ca în cazul limbilor străine, sau exemplificând alternativ implementări în cele două limbaje), evidențiind pe parcurs asemănările și deosebirile dintre acestea;

- succesiv, realizându-se o recapitulare a unor concepte, în celălalt limbaj, conform programei școlare.

Predarea elementelor de limbaj specifice Python și C++ în paralel oferă o abordare solidă, pe baza unui plan didactic eficient, având în vedere consolidarea conceptelor comune și susținerea învățării prin comparație (observând asemănări de logică și structură, diferențe de sintaxă, o înțelegere mai profundă a conceptelor de bază, cum ar fi gestionarea automată sau manuală a memoriei), evidențierea complementarității conceptuale (Python oferă o sintaxă simplă, ușor de citit, care permite elevilor să se concentreze pe concepte fără să fie pus accent pe detalii tehnice, iar C++ facilitează înțelegerea profundă a mecanismelor din spatele limbajului — tipuri de date, gestionarea memoriei, pointeri, compilare), formarea unei gândiri flexibile în programare, evidențiindu-se faptul că limbajul este doar un instrument, iar gândirea logică este universală, facilitându-se astfel adaptabilitatea la orice alt limbaj viitor.

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 XI-a, specializarea matematică-informatică, pentru clase cu predarea disciplinei informatică în regim intensiv, cu o abordare în paralel a celor două limbaje de programare

1. *Organizarea conceptuală a datelor. Modelul conceptual liniar – listă*
2. *Memorarea datelor și organizarea codului în limbaj de programare. Pointeri în C++*
3. *Strategii de rezolvare a problemelor. Metoda Backtracking*
4. *Strategii de rezolvare a problemelor. Metoda Programării dinamice*
5. *Organizarea conceptuală a datelor. Model conceptual rețea – graf*
6. *Strategii de rezolvare a problemelor. Algoritmi specifici pentru prelucrarea grafurilor*
7. *Organizarea conceptuală a datelor. Model conceptual ierarhic – arbore*
8. *Strategii de rezolvare a problemelor. Algoritmi specifici pentru prelucrarea arborilor*
9. *Organizarea conceptuală a datelor. Model conceptual obiectual*
10. *Memorarea datelor și organizarea codului în limbaj de programare. Programare orientată pe obiecte*
11. *Memorarea datelor și organizarea codului în limbaj de programare. Paradigme de bază în programare*

Pentru predarea metodei Backtracking se recomandă precizarea unor repere avute în vedere pentru rezolvarea problemelor (de exemplu, ce este o soluție dintre cele cerute, ce reprezintă un element al acesteia, care sunt condițiile de validare a unei valori propuse, care este condiția de obținere a unei soluții complete) și exemplificarea acestora pentru cazuri concrete. Se recomandă abordarea unor rezolvări cu aplicarea generării de elemente combinatoriale (de exemplu, permutări, aranjamente, combinații, produs cartezian), eventual cu restricții privind relațiile dintre elemente, probleme clasice în care o soluție se poate reprezenta ca o succesiune de perechi, în care poziția are o anumită semnificație (de exemplu, problema reginelor, în care o soluție este o succesiune de poziții pe tabla de șah, perechi linie-coloană, iar numărul de ordine în cadrul soluției reprezintă linia și valoarea corespunzătoare reprezintă coloana, problema colorării unei hărți, în care o soluție este o succesiune de perechi țară-culoare, iar numărul de ordine în cadrul soluției reprezintă țara și valoarea corespunzătoare reprezintă culoarea), probleme în care sunt preluate valori parțiale calculate la pașii anteriori (de exemplu, problema partițiilor unui număr, problema plății unei sume cu anumite tipuri de monede, problema partițiilor unei mulțimi), probleme de optim (de exemplu, problema unui comis-voiajor) ca model, apoi realizarea unor probleme asemănătoare care presupun adaptări, reinterpretări, extinderi.

Pentru Backtracking generalizat se recomandă rezolvarea unor probleme clasice model ca problema labirintului, problema acoperirii unei table de șah prin sărituri ale unui cal.

Pentru predarea metodei Programării dinamice se recomandă precizarea unor repere avute în vedere pentru rezolvarea problemelor, de exemplu, aranjarea problemei astfel încât să rezulte caracterul de proces cu mai multe etape, identificarea datelor calculate la etapele anterioare și necesare pentru pasul curent, respectiv a modului în care datele obținute la pasul curent pot interveni în alte calcule, găsirea unei relații de recurență între acestea, eliminarea recurenței prin calcularea și memorarea rezultatelor subproblemelor, și exemplificare acestora pentru cazuri concrete. Având în vedere că „în general, metoda programării dinamice se aplică problemelor de optimizare” (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 clasice ca model, de exemplu, abordarea unor probleme în care sunt memorate date simple (de exemplu, problema scării), liste (de exemplu, determinarea unei submulțimi de sumă dată a unei mulțimi cu valori distincte sau cu valori care se pot repeta, plata unei sume cu un număr minim de monede, problema celui mai lung subșir crescător), cu liste de liste (de exemplu, sume parțiale într-un tablou, problema rucsacului în varianta discretă, problema celui mai lung subșir comun a două șiruri), apoi realizarea unor probleme asemănătoare care presupun adaptări, reinterpretări, extinderi.

Pentru temele Model conceptual liniar – listă și Memorarea datelor și organizarea codului în limbaj de programare. Pointeri în C++, implementarea listelor înlanțuite în Python se face doar prin câteva exemple, utilizând clasa list, fiecare înlanțuire fiind dată prin poziția la care se află elementul următor din listă, iar în C++ se face doar prin alocare dinamică a memoriei.

Se recomandă ca în cadrul temelor Model conceptual rețea – graf și Metode de prelucrare a grafurilor reprezentările grafurilor (diagramă, matrice de adiacență, liste) să fie introduse succint, cu exemple intuitive (hartă, rețea de prieteni), iar accentul să fie pus ulterior pe utilizarea algoritmilor specifici, în mod structurat, urmând etape corespunzătoare pentru elaborarea programelor. Programa este flexibilă din punctul de vedere a ordinii de predare, astfel încât se pot preda, de exemplu, în paralel grafurile neorientate și orientate, apoi arborii, pentru a elimina redundanțele și a pune în evidență diferențele dintre acestea, sau se pot preda succesiv, pentru a fixa noțiunile specifice, de exemplu, grafuri neorientate, grafuri orientate, arbori sau grafuri neorientate, arbori, grafuri orientate. Astfel, pentru grafuri, se pot preda concepte de bază (definiții ale grafurilor neorientate/orientate, concepte ca adiacență, grade, lanț/drum, ciclu/circuit), metode de reprezentare și aplicații cu algoritmi de bază, de exemplu, pentru determinarea gradului unui nod/vârf, a nodurilor izolate, verificarea proprietății de lanț/drum pentru un șir de valori. După aceea, se pot prezenta metodele de parcurgere a grafurilor, grafuri asociate și tipuri de grafuri: grafuri complete, grafuri bipartite (definiții și utilizarea metodelor de parcurgere în aplicații), grafuri conexe (definiții și utilizarea metodelor de parcurgere în aplicații), grafuri tare conexe (definiții, algoritmi specializați pentru determinarea matricei drumurilor și utilizarea lor în aplicații), grafuri ponderate (definiții, algoritmi specializați pentru determinarea unor lanțuri/drumuri de cost minim și utilizarea acestora în aplicații), grafuri hamiltoniene și euleriene (definiții și teoreme, algoritmi specifici). La liceu nu se studiază grafuri cu bucle, grafuri infinite sau multigrafuri, considerându-se că oricare arc/muchie are extremități distincte și oricare două arce/muchii diferă prin cel puțin una dintre extremități. De remarcat că în lucrările de specialitate, algoritmul Roy-Floyd se regăsește și cu denumirea Floyd-Warshall.

În cadrul temei Model conceptual ierarhic – arbore, se recomandă să se pornească de la modele vizuale reale (arbore genealogic, ierarhie școlară, meniuri de site), iar după aceea să fie introduse concepte formale. Implementarea arborilor binari în Python se face utilizând clasa list, fiecare înlanțuire cu un nod ascendent/descendent fiind dată prin poziția la care se află acesta, iar în C++ se face doar prin alocare dinamică a memoriei.

Pentru tema Programare orientată pe obiecte, se recomandă evidențierea faptului că programarea orientată pe obiecte nu înseamnă doar „alt tip de sintaxă”, ci un mod de gândire în care problemele se modelează folosind entități (obiecte) și tipuri (clase), iar accentul se mută de la „ce face programul pas cu pas” la „cine (obiectul) face acțiunea”. Moștenirea evidențiază „specializarea” claselor, cu preluarea membrilor care definesc comportamente generale și aplicarea principiului polimorfismului pentru adaptarea acestora la specificul claselor derivate.

Proiectele interdisciplinare și învățarea pe baza unor aplicații care să conecteze teoria cu lumea reală (de exemplu, pentru grafuri, modelarea rețelelor sociale, a rutelor între orașe, pentru arbori, modelarea unor ierarhii organizaționale, a arborelui genealogic) 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.

Pentru tema Paradigme de bază în programare se vor prezenta succint conținuturile din programă, având în vedere diferențe și asemănări, prezentarea unui exemplu de cod într-un limbaj cunoscut și în alte limbaje pentru principalele aspecte evidențiate, punând în valoare facilitățile limbajelor/paradigmelor.

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,

Nume și prenume	Grad didactic/Titlu științific	Instituție de apartenență, localitate, județ
		Facultatea de Automatică și Calculatoare
NIȚU Alina	profesor, gradul didactic I	Liceul Teoretic „Ovidius”, Constanța, județul Constanța
NURLA Aidan	profesor, gradul didactic I	Colegiul Național Militar „Alexandru Ioan Cuza”, Constanța, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forțelor 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 Național Militar „Dimitrie Cantemir”, Breaza, județul Prahova
TEGLAȘ Maria	profesor, gradul didactic II	Colegiul Național 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

Anexa nr. 11

Programa școlară
pentru disciplina

Informatică

Clasa a XII-a

Curriculum de specialitate (CS)

*Filiera teoretică, profilul real, specializarea matematică-informatică,
clase cu predarea disciplinei informatică în regim intensiv*

Învățământ liceal

- 2026 –

Anexa nr. 11

Programa școlară pentru disciplina *Informatică*, clasa a XII-a, curriculum de specialitate (CS)

- *Filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei informatică în regim intensiv*

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 matematică - informatică, clase cu predarea disciplinei *informatică* în regim intensiv, alocarea orară este:

- clasa a IX-a – 4 ore/săptămână, dintre care două ore pentru studiu teoretic și două ore pentru activități practice;
- clasa a X-a – 4 ore/săptămână, dintre care două ore pentru studiu teoretic și două ore pentru activități practice;
- clasa a XI-a – 7 ore/săptămână, dintre care patru ore pentru studiu teoretic și trei ore pentru activități practice;
- clasa a XII-a – 7 ore/săptămână, dintre care patru ore pentru studiu teoretic și trei ore pentru activități practice.

Activitățile practice sunt desfășurate obligatoriu în laboratorul de informatică, iar în baza Ordinului ministrului educației și cercetării nr. 6873/2025, pentru filiera teoretică, profilul real, specializarea matematică-informatică, clase cu predarea disciplinei *informatică* în regim intensiv, activitățile practice se desfășoară cu colectivul de elevi organizat pe grupe.

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 (Ordinul 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);
- Cadrul de competențe de inteligență artificială pentru elevi (AI competency framework for students), UNESCO 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.

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 limbajelor de programare Python și 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. Limbajul 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 experiența 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 XII-a
CS 1.1. Identifică principalele caracteristici ale organizării datelor în cadrul unor modele conceptuale avansate - pentru proiectarea bazelor de date sau învățare automată, pentru a structura și accesa date în vederea prelucrării acestora
- identificarea elementelor principale ale unui model entitate-relație (entitate, atribut, relație, identificator unic, tipuri de identificatori unici - natural/artificial, simplu/compus) în contextul gestionării informațiilor despre cărți, elevi și împrumuturile realizate într-o bibliotecă școlară - recunoașterea tipurilor de cardinalitate (1:1, 1:M, M:M) dintr-o listă de exemple de relații între entități pentru gestionarea unui magazin online - identificarea termenilor specifici (supertip, subtip, arc, transferabilitate), în contextul gestionării unui sistem de analiză a rețelelor de transport, prin evidențierea tipurilor de trasee - identificarea caracteristicilor unui subtip al unei entități (moștenire de attribute, relații comune) pentru un sistem de gestionare a angajaților, prin evidențierea atributelor moștenite - recunoașterea etapelor de preprocesare a datelor (curățare, normalizare, codificare) într-un flux de analiză a datelor provenite din senzori medicali - identificarea conceptelor de bază (set de antrenare, set de testare, caracteristici, etichetă) într-un proiect practic ce presupune clasificarea automată a e-mailurilor în unele de tip spam/non-spam
CS 1.2. Identifică specificul, caracteristicile și etapele unor strategii de normalizare a modelului conceptual al unei probleme de gestiune
- enumerarea caracteristicilor datelor care corespund primei forme normale (FN1), în contextul unei baze de date pentru o bibliotecă, unde unele cărți au mai mulți autori - enumerarea caracteristicilor datelor care corespund celei de a doua forme normale (FN2), în contextul unei baze de date pentru o bancă, unde unele tabele au identificator compus - enumerarea caracteristicilor datelor care corespund celei de a treia forme normale (FN3), în contextul unei baze de date pentru o școală, unde pentru fiecare elev sunt memorate datele părinților
CS 1.3. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în învățarea automată
- enumerarea caracteristicilor învățării supervizate în contextul aplicării pentru soluționarea unor probleme practice pentru care avem acces la date etichetate de experți - enumerarea caracteristicilor învățării nesupervizate în contextul aplicării pentru soluționarea unor probleme practice pentru care nu avem acces la etichete - recunoașterea termenilor de bază (de exemplu, învățare supervizată, învățare nesupervizată, clasificare, regresie, clusterizare) în contextul unor situații practice din viața reală - identificarea caracteristicilor algoritmului K-means într-un context de clasificare a clienților după comportamentul de cumpărare - enumerarea pașilor principali ai unui algoritm de învățare bazat pe ajustarea ponderilor (feedforward și backpropagation) - reamintirea situațiilor în care rețelele neuronale pot fi aplicate (recunoaștere imagini, voce, text), prin exemple simple din viața cotidiană
CS 1.4. Identifică principalele comenzi SQL și ale limbajului de programare pentru a identifica soluția adecvată prelucrării datelor organizate în baze de date, în vederea rezolvării eficiente a problemelor informatice de gestiune
- identificarea tipurilor de comenzi SQL pentru gestionarea unei baze de date a unei biblioteci, prin clasificarea fiecărei comenzi în categoria corespunzătoare - descrierea structurii de bază a unei comenzi SELECT pentru interogarea unei baze de date a unui magazin online, prin enumerarea componentelor acesteia - recunoașterea comenzilor SQL pentru gestionarea bazei de date a unei clinici medicale, prin acțiuni corespunzătoare asupra tabelelor - enumerarea claselor și metodelor principale ale limbajului Python utilizate pentru conectarea la o bază de date pentru o aplicație de evidență a produselor - identificarea părților componente ale unui cod simplu de conectare la o bază de date (nume al clasei, metodă adecvată, parametri) pentru un script Python care accesează baza de date a unei clinici medicale - recunoașterea rolului bibliotecilor uzuale (sqlite3, mysql.connector) într-un fragment de cod Python pentru două aplicații diferite

Clasa a XII-a
CS 1.5. Identifică principalele elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în învățarea automată, în vederea rezolvării eficiente a problemelor informatice
- enumerarea bibliotecilor Python utilizate în învățarea automată (Pandas, Matplotlib, Scikit-learn) și precizarea scopului fiecăreia pentru înțelegerea ecosistemului de lucru cu date, specificând că Pandas gestionează date tabulare, Matplotlib vizualizări grafice și Scikit-learn algoritmi de machine learning - identificarea claselor principale (DataFrame, Series, LinearRegression) în fragmente de cod Python pentru recunoașterea structurilor de date într-un script de analiză a vânzărilor, distingând DataFrame ca tabel complet, Series ca o coloană și LinearRegression ca model de regresie liniară - recunoașterea tipurilor de grafice (linie, bare, histogramă, box-plot, scatter) generate de Matplotlib pentru interpretarea corectă a vizualizărilor dintr-un raport de analiză, identificând graficul de linie pentru evoluții temporale, bare pentru comparații categoricale, histogramă pentru distribuții, box-plot pentru statistici descriptive și scatter pentru corelații - recunoașterea modului de creare a array-urilor prin identificarea metodelor de bază din NumPy pentru operații cu liste de valori (np.array, np.zeros, np.ones, np.arange) - identificarea operațiilor uzuale de algebră liniară disponibile în NumPy, prin etichetarea fiecărei operații (înmulțire matriceală, transpunere, determinant, inversare)

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 XII-a
CS 2.1. Explică principiile de organizare a datelor în cadrul unor modele conceptuale avansate - pentru proiectarea bazelor de date sau învățare automată, pentru a le gestiona eficient
- explicarea diferenței dintre o entitate și o instanță a unei entități prin exemplul gestionării informațiilor despre elevi, profesori și discipline într-o școală - explicarea modului în care o relație M:M se transformă în două relații 1:M, de regulă barate, introducând o entitate de intersecție, prin exemplul unui sistem de notare a elevilor la fiecare disciplină, de-a lungul unui an de studiu - interpretarea unei diagrame simple entitate-relație și explicarea sensului simbolurilor utilizate (bare, opționalități, cardinalități) pentru un sistem de gestionare a comenzilor într-un restaurant, unde se analizează relațiile dintre clienți, comenzi și produse - explicarea diferenței dintre „supertip” și „subtip” în cadrul unui model conceptual pentru un sistem de gestionare a angajaților, prin evidențierea tipurilor și subtipurilor în cazul angajaților care lucrează part-time, full-time sau sezonieri - interpretarea unei diagrame entitate-relație care conține arce și formularea unei explicații a rolului acestora (relații alternative mutual exclusive) pentru un sistem de gestionare a transportului urban, evidențiind tipul de transport care poate fi ales la un moment dat (troleu, autobuz, metrou) - explicarea diferenței dintre date brute și date preprocesate folosind ca exemplu un set de date despre pacienți (unde unele valori lipsesc sau sunt exprimate în unități de măsură diferite) - ilustrarea prin exemple a modului în care structura datelor influențează performanța modelului într-un caz practic de predicție a prețului apartamentelor - explicarea, pe baza unui exemplu din domeniul sănătății, a modului în care datele incomplete sau neomogene pot afecta performanța unui model de diagnostic automat
CS 2.2. Explică etapele unor strategii de normalizare a modelului conceptual al unei probleme de gestiune
- explicarea importanței normalizării unei baze de date pentru un sistem de gestionare a comenzilor unui magazin online, prin evidențierea reducerii redundanțelor, a prevenirii inconsistențelor și a simplificării prelucrării datelor - explicarea diferențelor dintre prima, a doua și a treia formă normală, pentru un sistem de gestionare a comenzilor unui magazin online - interpretarea unui model conceptual dat pentru un sistem de gestionare a vânzărilor unui magazin, prin evidențierea respectării formelor normale
CS 2.3. Explică etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în învățarea automată
- explicarea principiilor învățării nesupervizate și clusterizării pe baza unui exemplu de grupare a utilizatorilor unei platforme video în funcție de preferințe - explicarea diferențelor dintre clasificare și regresie pe baza unui set de date cu vânzări lunare (regresie) versus tipul clientului (clasificare) - ilustrarea modului în care algoritmul KNN utilizează vecinii apropiați pentru a clasifica un element într-un scenariu simplu de clasificare a florilor (setul Iris) - explicarea modului în care alegerea atributelor poate afecta performanțele unui algoritm de învățare automată utilizat pentru rezolvarea unei probleme de predicție a prețului unui apartament - explicarea modului de funcționare a unui neuron artificial, prin reprezentarea grafică a intrărilor, ponderilor și funcției de activare - explicarea modului în care o rețea perceptron procesează informația și își ajustează conexiunile pentru a învăța din erori

Clasa a XII-a	
CS 2.4. Explică rolul componentelor principalelor comenzi SQL și ale limbajului de programare pentru a identifica soluția adecvată prelucrării datelor organizate în baze de date, în vederea rezolvării eficiente a problemelor informatice de gestiune	
- explicarea diferenței dintre comenzile DDL și DML pentru gestionarea bazei de date a unei școli, prin evidențierea acțiunilor corespunzătoare asupra structurii, respectiv asupra conținutului tabelelor - explicarea utilizării funcțiilor agregate într-o interogare pentru analiza vânzărilor dintr-un supermarket, prin rezultatul obținut în urma aplicării lor - interpretarea unui exemplu de interogare SQL și explicarea rezultatului pe baza datelor afișate pentru o bază de date cu angajați și departamente, prin analizarea pas cu pas a fiecărei clauze - explicarea rolului unei vederi (view) și a avantajelor utilizării acesteia pentru simplificarea accesului la date - interpretarea unei secvențe SQL care folosește comenzi precum COMMIT și ROLLBACK, în contextul unei tranzacții bancare de transfer, prin explicarea efectului fiecărei comenzi asupra bazei de date - explicarea, în propriile cuvinte, a diferenței dintre obiectul de tip Connection și cel de tip Cursor pentru înțelegerea arhitecturii de lucru cu baze de date în Python, evidențiind faptul că obiectul de tip Connection reprezintă legătura fizică la baza de date (sesiunea de comunicare), iar obiectul de tip Cursor este cel care execută comenzile SQL și gestionează rezultatele, ilustrând analogia cu o carte (Connection) și un semn de carte (Cursor) care marchează poziția curentă - descrierea pașilor necesari pentru realizarea unei conexiuni la o bază de date și executarea unei interogări pentru un script Python care accesează o bază de date SQLite cu informații despre produse, detaliind secvența: importarea bibliotecii (sqlite3), crearea conexiunii cu connect(), crearea obiectului Cursor cu cursor(), executarea interogării cu execute(), preluarea rezultatelor cu fetchall() sau fetchone(), și închiderea resurselor cu close() - interpretarea unui fragment de cod Python care utilizează obiecte de tip Connection și Cursor, prin explicarea secvenței de operații efectuate pentru a extrage date din tabela Comenzi, identificând fiecare linie de cod, rolul acesteia (conectare, executare, preluare date, închidere), ordinea de executare și modul în care datele sunt transferate de la baza de date în vederea prelucrării lor	
CS 2.5. Explică rolul principalelor elemente ale limbajului de programare utilizate pentru prelucrarea datelor în învățarea automată, în vederea rezolvării eficiente a problemelor informatice	
- explicarea modului în care un obiect DataFrame reprezintă datele sub formă de tabel pentru înțelegerea structurii de date Pandas, prin organizarea informațiilor - explicarea modului în care array-urile NumPy și DataFrame-urile Pandas stochează și manipulează datele pentru alegerea structurii potrivite în analiza datelor - interpretarea unui fragment de cod care construiește o regresie liniară simplă cu Scikit-learn pentru predicția prețurilor unor apartamente pe baza suprafeței, prin antrenarea unui model pe datele disponibile - explicarea diferențelor dintre operațiile cu liste Python și array-uri NumPy, prin descrierea avantajelor utilizării NumPy în învățare automată - explicarea rezultatelor operațiilor care implică broadcasting în NumPy, pentru înțelegerea adunării unui scalar la o matrice sau înmulțirea unui vector linie cu unul coloană	

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

Clasa a XII-a	
CS 3.1. Utilizează modul de organizare și prelucrare a datelor în cadrul unor modele conceptuale avansate – la proiectarea bazelor de date sau la învățare automată, pentru a rezolva probleme de gestionare a datelor	
- aplicarea regulilor pentru transformarea unei relații M:M prin exemplul unui sistem de închiriere filme, unde se utilizează regula introducerii unei entități de intersecție, pentru împrumuturi - completarea unei diagrame entitate-relație parțiale cu marcatorii adecvați pentru atribute, respectând regulile de identificare unică pentru un sistem de gestiune a vânzărilor, care conține entitățile PRODUS, CLIENT și COMANDĂ - completarea unei diagrame entitate-relație parțiale cu supertipuri și subtipuri corespunzătoare unui scenariu dat, prin exemplul mijloacelor de transport pentru persoane - aplicarea regulilor de reprezentare pentru plasarea unui arc pe trei relații posibile, respectând regula de exclusivitate prin exemplul unui sistem de gestionare a rezervărilor la un hotel - aplicarea unor operații de augmentare a datelor (de exemplu, rotirea sau scalarea imaginilor) pentru un set de imagini cu fructe - aplicarea operației de normalizare a valorilor numerice ale unui set de date cu puține exemple provenit dintr-o bază de date cu indicatori economici, manual, prin calculul valorilor normalizate pe baza valorilor minime și maxime, pentru a aduce toate valorile pe aceeași scară - aplicarea unor criterii pentru separarea unui set de date despre rezultate școlare în seturi de antrenare și testare, în vederea asigurării existenței unor date noi care să fie utilizate pentru evaluarea unui algoritm de învățare automată	

Clasa a XII-a	
CS 3.2. Aplică strategii de normalizare a modelului conceptual al unei probleme de gestiune	
- aplicarea strategiei pentru obținerea primei forme normale (FN1) pentru un sistem de gestionare a comenzilor unui magazin online, prin tratarea adecvată a fiecărui atribut al unei entități care nu are un caracter atomic, fiind format din mai multe componente care sunt utilizate independent - aplicarea strategiei pentru obținerea celei de a doua forme normale (FN2) pentru un model conceptual al unei baze de date, prin tratarea adecvată a dependențelor unor attribute de componente ale identificatorului unic compus, pentru un sistem de gestiune a activităților nonformale dintr-o școală - aplicarea strategiei pentru obținerea celei de a treia forme normale (FN3) a unui model conceptual al unei baze de date, prin tratarea adecvată a dependențelor unor attribute de alte attribute diferite de identificatorul unic, cu respectarea integrității datelor pentru un sistem de gestiune a angajaților și departamentelor unei companii	
CS 3.3. Aplică algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în învățarea automată	
- aplicarea algoritmului K-means pe un set de date cu coordonate 2D reprezentând sediile a 15 magazine dintr-un oraș în vederea identificării a trei zone optime pentru amplasarea depozitelor de distribuție, urmărind cum centroizii se deplasează progresiv către pozițiile finale și cum magazinele sunt reatribuite la fiecare iterație - aplicarea algoritmului de regresie liniară pentru soluționarea unei probleme de predicție a prețurilor caselor pornind de la un set de date ce conține cinci proprietăți pentru care cunoaștem suprafața și prețul - aplicarea algoritmului de clasificare KNN pentru un set de date simplu cu fructe (mere, portocale, banane) descrise prin greutate și diametru - simularea funcționării unui neuron artificial, utilizând valori numerice simple pentru a calcula ieșirea pe baza intrărilor și ponderilor - aplicarea principiilor rețelelor perceptron pentru rezolvarea unei probleme simple de clasificare (recunoașterea formelor geometrice)	
CS 3.4. Utilizează principalele comenzi SQL și ale limbajului de programare pentru a identifica soluția adecvată prelucrării datelor organizate în baze de date, în vederea rezolvării eficiente a problemelor informatice de gestiune	
- aplicarea unei interogări SQL pentru baza de date a unui liceu, pentru afișarea elevilor cu media peste 9, în ordine alfabetică - aplicarea unei comenzi JOIN pentru obținerea unui raport școlar, prin extragerea informațiilor din două tabele relaționate - aplicarea unei funcții de conversie într-o interogare pentru a transforma un tip de date numeric în șir de caractere prin generarea de etichete cu prețuri într-un magazin - aplicarea comenzilor GRANT și REVOKE pentru gestionarea drepturilor de acces ale diferiților utilizatori, într-un sistem de rezervări hoteliere - utilizarea comenzilor SAVEPOINT, ROLLBACK și COMMIT, pentru a realiza tranzacții cu revenire la un punct stabilit, pentru procesarea comenzilor unui magazin online - utilizarea instrumentelor limbajului Python pentru conectarea la o bază de date SQLite și afișarea conținutului unei tabele pentru vizualizarea produselor dintr-un magazin, prin crearea conexiunii, a cursorului, executarea interogării și afișarea rezultatelor - utilizarea metodelor execute() și fetchall() și a comenzii SELECT pentru preluarea și afișarea rezultatelor unei interogări în vederea extragerii tuturor comenzilor dintr-o perioadă specificată - utilizarea metodelor execute(), close() și commit() și a comenzii INSERT INTO, pentru a insera date într-o tabelă, folosind conexiunea și cursorul corespunzător și închizând conexiunea corect, în contextul adăugării unui client nou în tabela Clienti	
CS 3.5. Utilizează principalele elemente ale limbajului de programare pentru prelucrarea datelor în învățarea automată, în vederea rezolvării eficiente a problemelor informatice	
- utilizarea metodelor din Pandas pentru citirea unui set de date dintr-un fișier Note.csv și afișarea valorilor statistice de bază (medie, minim, maxim, deviație standard) în contextul analizei situației școlare - aplicarea metodelor din Matplotlib pentru reprezentarea grafică a datelor privind vânzările lunare ale unui magazin, prin realizarea unor grafice adecvate (de exemplu, grafic liniar, diagramă cu bare) - utilizarea metodelor fit() și predict() ale clasei LinearRegression din Scikit-learn pentru a antrena un model de regresie liniară și realizarea unei predicții pentru consumul de combustibil pe baza greutateii mașinii și a distanței parcurse - aplicarea funcțiilor și operațiilor din biblioteca NumPy pentru rezolvarea unui sistem de ecuații liniare, în vederea determinării prețurilor unor produse pe baza unui model de cost - aplicarea operațiilor numerice din biblioteca NumPy pentru implementarea explicită a normalizării (Min-Max) și standardizării (Z-score) unui set de date, prin calculul minimului, maximumului, mediei și deviației standard, în vederea construirii unei matrice de caracteristici pentru antrenarea unui model de învățare automată	

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 XII-a
CS 4.1. Analizează caracteristicile, modul de prelucrare, avantajele și dezavantajele utilizării unor modele conceptuale avansate – pentru proiectarea bazelor de date sau pentru învățare automată, pentru a alege structura adecvată în rezolvarea unor probleme concrete

Clasa a XII-a	
- analizarea unui model conceptual dat, corespunzător activității de programare a cursanților dintr-o școală de conducători auto, pentru identificarea corespondenței dintre scenariu și entitățile și relațiile modelului - analizarea unui model entitate-relație existent pentru a identifica erorile de cardinalitate pentru un sistem de gestionare a rezervărilor la un hotel, prin examinarea relațiilor dintre clienți, camere și rezervări - analizarea unui model entitate-relație dat pentru a identifica erorile de redundanță pentru un sistem de gestionare a comenzilor într-un magazin online - analizarea unei diagrame entitate-relație pentru a detecta și corecta relații redundante sau incorect modelate pentru un sistem de gestionare a angajaților, cu subtipuri care omit atributele moștenite sau relațiile comune - compararea a două variante de model conceptual pentru același scenariu prin identificarea diferențelor în utilizarea arcelor sau a transferabilității pentru un sistem de gestionare a proiectelor - analizarea structurii unui set de date real (număr de atribute, tipuri de variabile, valori lipsă), pentru identificarea pașilor de preprocesare necesari - compararea diferitelor metode de transformare a datelor (scalare min-max vs. standardizare), pentru identificarea contextului potrivit pentru fiecare, pornind de la o problemă din viața reală - analizarea corelațiilor dintre variabile într-un set de date despre obiceiuri alimentare și starea de sănătate, pentru a identifica atributele relevante pentru predicția finală	
CS 4.2. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de normalizare a modelului conceptual al unei probleme de gestiune	
- analizează avantajele unui model conceptual ce respectă prima formă normală (FN1), comparativ cu un model care nu respectă această formă, în contextul obținerii orașului de domiciliu al unei persoane, pentru cazul în care acesta este inclus în adresă (utilizarea unei părți a unui atribut) și pentru cazul în care el reprezintă un câmp individual (utilizarea acestuia ca o valoare atomică) - detectarea dependențelor funcționale dintre atribute și clasificarea lor pentru un sistem de gestiune a angajaților și proiectelor dintr-o companie, prin identificarea dependențelor parțiale și a celor tranzitive - compararea a două modele conceptuale ale aceleiași situații prin evidențierea celei care respectă principiile normalizării pentru un sistem de gestionare a comenzilor și clienților unui magazin de electronice	
CS 4.3. Analizează comportamentul, aplicabilitatea, avantajele și dezavantajele utilizării unor algoritmi specializați pe clase de probleme pentru învățarea automată	
- analizarea diferențelor dintre învățarea supervizată și cea nesupervizată, prin completarea unui tabel comparativ cu exemple concrete (de exemplu, predicția notelor vs. gruparea elevilor după interese) - compararea performanței algoritmilor K-means, regresie liniară și KNN, prin aplicarea fiecăruia pe seturi de date adecvate problemelor pe care le pot rezolva, din perspectiva metricii specifice fiecărui tip de problemă (de exemplu, coeficientul Silhouette pentru K-means, eroarea medie pătratică pentru regresie liniară, acuratețea sau matricea de confuzie pentru KNN) - interpretarea grafică a rezultatelor obținute de diferiți algoritmi, identificând modelele de date și abaterile față de tendință - analizarea unui caz (de exemplu, recomandarea de filme) pentru a identifica tipul de algoritm de învățare automată cel mai potrivit și justificarea alegerii în funcție de natura problemei și a datelor disponibile - analizarea structurii unei rețele neuronale, prin identificarea rolului fiecărui strat (intrare, ascuns, ieșire) într-o schemă grafică - compararea performanței unei rețele perceptron cu o rețea cu mai multe straturi, pe același set simplu de date	
CS 4.4. Analizează aplicabilitatea, avantajele și dezavantajele utilizării comenzilor SQL și a elementelor specifice limbajului de programare, în vederea rezolvării eficiente a problemelor informatice de gestiune	
- examinarea unei comenzi SQL pentru o interogare care extrage comenzile dintr-un sistem de gestiune a unui magazin online, prin identificarea eventualelor erori de sintaxă sau de logică - compararea a două interogări care obțin același rezultat, dar folosesc metode diferite (JOIN vs. subinterogare) pentru afișarea produselor comandate de un client specific dintr-un sistem e-commerce, prin evaluarea avantajelor și dezavantajelor fiecărei abordări - examinarea părților componente ale unei interogări complexe pentru un raport de vânzări lunar, prin explicarea rolului fiecăreia - examinarea corectitudinii unui set de comenzi SQL pentru actualizarea stocului după vânzări, prin identificarea punctelor vulnerabile - compararea a două metode de control al accesului (prin GRANT și prin utilizatori dedicați) pentru un sistem de salarizare, prin identificarea avantajelor fiecărei metode - analizarea unui cod Python care realizează o conexiune la o bază de date și identificarea posibilelor erori (lipsa commit(), conexiune neînchisă) pentru un script care actualizează prețurile produselor, verificând dacă lipsește commit() după UPDATE (modificările nu se salvează), dacă conexiunea rămâne deschisă (resurse blocate) sau dacă nu există tratare de excepții - compararea a două modalități de executare în Python a comenzilor SQL (prin execute() și prin executemany()) și determinarea cazului potrivit fiecăreia pentru inserarea de date în tabela Elevi, stabilind că execute() este eficient pentru o	

Clasa a XII-a
singură înregistrare, iar <code>executemany()</code> este optim pentru inserarea mai multor înregistrări dintr-o listă, reducând numărul de apeluri - descompunerea unui exemplu complet (creare, inserare, interogare, închidere) pentru a evidenția ordinea corectă a apelurilor de metode pentru un script (de exemplu, pentru gestiunea unei biblioteci), secvențializând: <code>connect()</code> → <code>cursor()</code> → <code>execute(CREATE TABLE)</code> → <code>execute(INSERT)</code> → <code>commit()</code> → <code>execute(SELECT)</code> → <code>fetchall()</code> → <code>close()</code>, identificând dependențele dintre pași
CS 4.5. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor elemente ale limbajului de programare, pentru a identifica soluția adecvată prelucrării datelor în învățarea automată, în vederea rezolvării eficiente a problemelor informatice - analizarea unui program Python care citește, curăță și vizualizează datele pentru a identifica ordinea corectă a pașilor de prelucrare pentru un script ce analizează tranzacțiile comerciale - compararea rezultatelor obținute cu funcția <code>describe()</code> din Pandas cu valorile calculate manual pe baza operațiilor de bază cu liste pentru statisticile de bază (medie, minim, maxim, deviație standard) pe un set de date cu temperaturi, verificând concordanța calculelor - analizarea unui grafic complex pentru a explica modul în care datele sunt combinate și afișate în Matplotlib pentru un panou de control cu grafice multiple despre performanța elevilor - compararea diferitelor metode NumPy pentru aceeași operație, prin investigarea diferențelor de comportament și performanță pentru înmulțirea matricilor în diverse scenarii - analizarea structurii unui array multidimensional, prin determinarea axelor și dimensiunilor pentru operații de agregare

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

Clasa a XII-a
CS 5.1. Argumentează alegerile făcute în cadrul proiectării unor modele conceptuale avansate - pentru proiectarea bazelor de date sau pentru învățare automată, în vederea rezolvării unor probleme informatice concrete - evaluarea unui model entitate-relație din perspectiva respectării regulilor de identificare unică a instanțelor pentru un sistem de gestionare a închirierilor de mașini - evaluarea unui model entitate-relație din punctul de vedere al rezolvării adecvate a relațiilor M:M, pentru un sistem de gestionare a unei biblioteci digitale - justificarea deciziei de a folosi o relație ierarhică în locul uneia recursive, pentru un sistem de gestionare a structurii organizaționale a unei companii, prin analiza relației dintre angajați și manageri - evaluarea corectitudinii utilizării arcelor într-un model entitate-relație dat, pentru un sistem de gestionare a transportului urban - justificarea alegerii unui anumit mod de reprezentare a datelor sub formă de subtipuri sau entități separate, pentru un sistem de gestionare a angajaților - evaluarea calității unui set de date privind tranzacțiile bancare (din punctul de vedere al completitudinii și relevanței) înainte de utilizarea într-un algoritm de învățare automată - evaluarea impactului unei variante de normalizare asupra preciziei unui model de învățare automată aplicat pe un set de date despre diabet - argumentarea deciziei de a folosi codificarea de tip one-hot în locul asocierii unei etichete pentru variabile categorice ce descriu tipul unei mașini (sedan, SUV), în funcție de algoritmul de învățare automată utilizat
CS 5.2. Evaluează corectitudinea și eficiența modelelor conceptuale ale unei probleme de gestiune din punctul de vedere al normalizării - evaluarea corectitudinii procesului de normalizare în cadrul bazelor de date aplicat pentru un sistem de gestiune al comenzilor și clienților unui magazin - argumentarea avantajelor și dezavantajelor utilizării formelor normale pentru un sistem de gestionare a comenzilor și stocurilor unui depozit, prin evidențierea avantajelor precum reducerea redundanțelor și creșterea consistenței datelor și a dezavantajelor precum creșterea numărului de tabele și complexitatea interogărilor - justificarea păstrării sau eliminării unei relații dintr-un model ERD în funcție de impactul asupra integrității și redundanței datelor pentru un sistem de gestionare a comenzilor și produselor unui magazin - identificarea atributelor care provoacă denormalizarea într-o bază de date pentru gestionarea vânzărilor dintr-un magazin, prin evidențierea atributelor care se repetă sau depind de alte atribute decât cheia primară
CS 5.3. Evaluează corectitudinea și eficiența soluțiilor cu algoritmi specializați pe clase de probleme pentru învățare automată - evaluarea performanței unui model de regresie liniară pentru estimarea consumului de combustibil pe baza greutateii mașinii, prin compararea valorilor reale cu cele prezise și calcularea erorii medii (Mean Absolute Error sau Mean Squared Error) - argumentarea alegerii algoritmului potrivit (K-means, regresie liniară sau KNN) în funcție de scopul analizei și tipul datelor

Clasa a XII-a	
- formularea concluziilor privind precizia și utilitatea modelului KNN pentru clasificarea tipurilor de fructe pe baza caracteristicilor lor (greutate, culoare, diametru), folosind o valoare concretă a lui k (de exemplu, $k=3$), prin compararea valorilor reale și prezise obținute - argumentarea alegerii tipului de rețea sau a algoritmului de învățare potrivit (de exemplu, perceptron) pentru o anumită problemă practică - aprecierea limitelor modelelor de rețele neuronale, prin identificarea situațiilor în care acestea pot produce erori sau rezultate imprevizibile	
CS 5.4. Evaluează corectitudinea și eficiența secvențelor de cod sau aplicațiilor care utilizează principalele comenzi SQL și ale limbajului de programare pentru prelucrarea datelor organizate în baze de date, în vederea rezolvării eficiente a problemelor informatice de gestiune	
- evaluarea eficienței unei interogări SQL pentru un raport al unui sistem bancar care afișează clienții cu tranzacții suspecte, prin analizarea performanței interogării actuale și propunerea de eventuale versiuni îmbunătățite - validarea respectării principiilor de integritate referențială pentru baza de date a unei universități, prin examinarea consecințelor ștergerii unor date asupra înregistrărilor dependente - justificarea alegerii unui anumit tip de funcție (agregată vs. scalară) pentru obținerea rezultatului dorit în contexte diferite dintr-un sistem de resurse umane prin calcularea salariului mediu pe departament versus formatarea numelui complet al angajatului - argumentarea alegerii între o vedere și o interogare directă, în funcție de cerințele de securitate și performanță pentru afișarea datelor - justificarea folosirii comenzilor COMMIT sau ROLLBACK pentru un scenariu de lucru colaborativ prin actualizări simultane într-un sistem de rezervări - evaluarea eficienței unui program Python care lucrează cu baze de date, din punctul de vedere al timpului de executare și al gestionării resurselor pentru o aplicație de import masiv de tranzacții, analizând dacă executemany() este folosit corect, conexiunea se deschide/închide repetat ineficient și dacă fetchall() consumă prea multă memorie pentru seturi mari de date - argumentarea alegerii unei metode de conectare (sqlite3 local vs. mysql.connector extern) în funcție de scopul aplicației pentru proiectarea unui sistem de evidență, justificând sqlite3 pentru aplicații desktop single-user cu portabilitate (de exemplu, catalog personal) sau mysql.connector pentru aplicații multi-user cu acces concurent (de exemplu, sistem școlar) - justificarea utilizării blocurilor try-except-finally pentru gestionarea erorilor și închiderea sigură a conexiunilor pentru un modul de actualizare a datelor critice, demonstrând că try asigură execuția codului riscant, except prinde erori de conexiune/sintaxă SQL fără a opri programul, iar finally garantează close() chiar dacă apar excepții	
CS 5.5. Evaluează corectitudinea și eficiența aplicațiilor care utilizează elemente specifice de limbaj de programare pentru prelucrarea datelor în învățarea automată, în vederea rezolvării eficiente a problemelor informatice	
- evaluarea performanței unui model de regresie liniară pentru predicția prețurilor imobiliare, prin compararea valorilor reale și prezise și calcularea erorii medii pătratică (Mean Squared Error) folosind mean_squared_error din Scikit-learn, pentru setul de date de test - argumentarea alegerii tipului de reprezentare grafică (de exemplu, diagramă de dispersie vs. histogramă) în funcție de tipul de date pentru vizualizarea relației notă-timp de studiu, pentru un raport privind notele și timpul de studiu dintr-o platformă de cursuri online - justificarea utilizării unei metode de curățare a datelor într-un set concret privind numărul de exerciții rezolvate de elevi pentru fiecare temă, cu valori incomplete corespunzătoare zilelor în care elevii au lipsit, prin evaluarea impactului asupra rezultatelor - argumentarea alegerii NumPy față de alte biblioteci pentru operații de algebră liniară în contextul învățării automate, prin justificarea criteriilor de performanță - evaluarea limitărilor NumPy în procesarea seturilor mari de date, prin observarea situațiilor în care operațiile devin lente sau consumul de memorie crește semnificativ și identificarea nevoii de divizare a datelor	

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

Clasa a XII-a	
CS 6.1. Proiectează reprezentări personalizate ale unor modele conceptuale avansate - pentru baze de date sau adecvate pentru învățare automată, în vederea rezolvării unor probleme	
- proiectarea unui model entitate-relație personalizat pentru un sistem de gestionare a rezervărilor la un hotel, prin determinarea entităților și a relațiilor dintre acestea - proiectarea unui model entitate-relație personalizat pentru un sistem de gestionare a istoricului comenzilor și prețurilor într-un magazin online, pentru a asigura funcționarea corectă a bazei de date - îmbunătățirea unui model entitate-relație existent, cu justificarea modificărilor aduse pentru un sistem de gestionare a rezervărilor la un restaurant - proiectarea unei diagrame entitate-relație care include tipuri și subtipuri pentru un sistem de gestionare a unei școli - crearea unei versiuni alternative a unui model entitate-relație existent, prin optimizarea cu arce, pentru gestionarea traficului	

Clasa a XII-a	
- schematizarea fluxului de prelucrare a datelor într-un proiect de învățare automată (colectare, curățare, transformare, antrenare) utilizat pentru detecția automată a unor probleme ale plămânilor, pornind de la imagini medicale de tip CT - proiectarea unui model conceptual al unui set de date (atribute, relații, etichete) pentru o problemă de predicție (de exemplu, performanța elevilor, vânzări, temperaturi) - crearea unui set de date sintetic care simulează valorile unor senzori de mediu (temperatură, umiditate, presiune) și care poate fi folosit pentru antrenarea unui model de predicție	
CS 6.2. Proiectează soluții cu aplicarea personalizată a strategiilor pentru normalizarea modelului conceptual al unei probleme de gestiune	
- proiectarea unui model conceptual pentru o aplicație de gestiune prin respectarea celor trei forme normale, pentru un magazin de electronice - elaborarea unui model conceptual normalizat pentru un sistem de gestionare a închirierilor de mașini, prin aplicarea fiecărei etape de normalizare - crearea unei reprezentări grafice (ERD) care demonstrează trecerea progresivă prin 1NF, 2NF și 3NF pentru un sistem de gestionare a rezervărilor la un hotel, prin prezentarea ERD inițial, apoi ERD după aplicarea fiecărei forme de normalizare	
CS 6.3. Proiectează soluții cu aplicarea personalizată a algoritmilor specializați pe clase de probleme pentru învățare automată	
- proiectarea unui mini-experiment de învățare automată, pornind de la un set de date colectat de elevi (de exemplu, obiceiuri de lectură, preferințe muzicale) - crearea unui model simplu de clasificare sau regresie utilizând aplicații educaționale (Orange Data Mining, Teachable Machine, Excel) - elaborarea unui scurt raport digital care descrie etapele aplicării algoritmului ales (pregătirea datelor, antrenarea modelului, interpretarea rezultatelor) - proiectarea unui experiment pentru testarea diferitelor rate de învățare în antrenarea perceptronului, prin combinarea rezultatelor într-un raport vizual pentru problema clasificării pacienților cu anemie pe baza parametrilor sanguini	
CS 6.4. Implementează secvențe de cod sau aplicații care integrează în mod personalizat comenzi SQL și elemente ale limbajului de programare pentru prelucrarea datelor organizate în baze de date, în vederea rezolvării eficiente a problemelor informatice de gestiune	
- crearea tabelelor pentru un sistem de gestiune a unei bibliotecii, prin scrierea comenzilor DDL, respectând ordinea adecvată de definire a acestora - elaborarea unui set de comenzi SQL pentru analiza datelor unui sistem de rezervări hoteliere, prin generarea de rapoarte destinate managementului - crearea unei interogări avansate cu JOIN, GROUP BY, funcții agregate și condiții în HAVING, pentru a genera un raport personalizat pentru un lanț de restaurante prin identificarea sucursalelor cu vânzări peste medie - crearea unei baze de date care utilizează vederi, secvențe și indecși, pentru un sistem de rezervări - elaborarea unui set de comenzi SQL care gestionează tranzacții (BEGIN TRANSACTION, SAVEPOINT, COMMIT, ROLLBACK) și drepturi de acces pentru sistemul de procesare comenzi e-commerce, prin implementarea tranzacțiilor și tratare completă a erorilor - proiectarea unei aplicații Python care permite introducerea, modificarea și ștergerea datelor dintr-o bază de date (CRUD) pentru gestionarea inventarului unui magazin, implementând funcții separate create_produș(), read_produș(), update_pret() și delete_produș(), fiecare gestionând conexiunea, executând comanda SQL corespunzătoare (INSERT/SELECT/UPDATE/DELETE) și validând datele de intrare - elaborarea unui modul Python propriu care gestionează conexiunile și oferă funcții reutilizabile pentru execuția interogărilor pentru standardizarea accesului la baza de date în mai multe scripturi, creând clasa DatabaseManager cu metode connect(), execute_query(), execute_update() și close(), încapsulând logica de conexiune și tratare erori pentru reutilizare în diferite module - dezvoltarea unei aplicații complexe care folosește mai multe clase și metode (Connection, Cursor, commit, rollback) pentru operarea sigură a tranzacțiilor pentru un sistem bancar de transfer fonduri între conturi, implementând tranzacții atomice care debitează un cont și creditează altul, folosind commit() doar dacă ambele operații reușesc sau rollback() la eroare pentru menținerea consistenței datelor	
CS 6.5. Implementează aplicații care integrează în mod personalizat elemente specifice de limbaj de programare pentru prelucrarea datelor în învățarea automată, în vederea rezolvării eficiente a problemelor informatice	
- proiectarea unei aplicații Python complete care încarcă un fișier .csv, curăță datele, calculează statistici descriptive și afișează grafice multiple cu Matplotlib pentru analiza performanței sportive a elevilor dintr-un club sportiv, prin implementarea unui flux complet de activități - elaborarea unui mini-model de învățare automată pentru predicția mediei finale pe baza notelor la teste, utilizând clasele LinearRegression sau DecisionTreeClassifier din Scikit-learn și împărțirea datelor în seturi de antrenare și testare prin funcția train_test_split() - crearea unui mini-proiect de învățare automată care include preprocesarea datelor, documentând fiecare etapă, pentru un set de date privind prețurile locuințelor, având în vedere curățarea și completarea valorilor lipsă, codificarea variabilelor	

Clasa a XII-a

categorice, normalizarea valorilor numerice și antrenarea unui model simplu, cu documentarea fiecărei etape într-un notebook Python

- *proiectarea unui pipeline complet de preprocesare a datelor folosind exclusiv NumPy, prin combinarea operațiilor de curățare, transformare și normalizare*
- *modificarea unui cod, care folosește liste Python pentru a memora coordonatele a 1000 de puncte date și calculează distanțele euclidiene pentru toate perechile de puncte, pentru a îl optimiza cu operații vectorizate, utilizând biblioteca NumPy*

CONȚINUTURI ALE ÎNVĂȚĂRII

Clasa a XII-a

Domenii de conținut	Conținuturi
1. Organizarea conceptuală a datelor	1.1. Modelul conceptual relațional – entitate-relație - concepte de bază și caracteristici ale modelului conceptual entitate-relație: entitate (cu atribute opționale, obligatorii, volatile, respectiv cu identificatori unici, tipuri de identificatori unici – simplu/compus, natural/artificial), instanță, relație (opționalitate, cardinalitate, tipuri de relații din punctul de vedere al cardinalității, relații barate, relații ierarhice, relații recursive), supertip și subtip, transferabilitate, arc; - repere pentru identificarea entităților și reprezentarea convențională a diagramei entitate-relație (ERD), rezolvarea relațiilor M:M (mai mulți la mai mulți), reprezentarea istoricului datelor; - concepte de bază și caracteristici ale modelului fizic: tabelă (cu câmpuri, tipuri de date, cheie primară, cheie externe, înregistrări, constrângeri, reguli de integritate - de entitate și referențială); - repere pentru transformarea modelului conceptual în model fizic (mapare). 1.2. Modele conceptuale de organizare a datelor folosite în învățarea automată - concepte de bază și caracteristici ale învățării automate: date, atribute, etichete, model, antrenare (învățare), testare, predicție; - caracteristici ale seturilor de date utilizate în învățarea automată: tipuri de date, date categorice, date complete/lipsă, dimensiune și calitate a datelor; - repere pentru transformarea datelor în formate utilizabile de către algoritmi de inteligență artificială (normalizare, scalare, augmentare, codificare a datelor, tratare a datelor lipsă, organizarea datelor în fișiere sau tabele pentru prelucrare automată).
2. Strategii de rezolvare a problemelor	2.1. Forme normale pentru proiectarea modelului conceptual al unei probleme de gestiune - caracteristici și avantaje ale primelor trei forme normale (FN1, FN2, FN3) și repere pentru utilizarea acestora în proiectarea modelului conceptual al unei probleme de gestiune. 2.2. Algoritmi utilizați în învățare automată - concepte de bază și caracteristici: învățare nesupervizată, clusterizare; - caracteristici ale algoritmului K-means, pentru învățarea nesupervizată; - concepte de bază și caracteristici: învățare supervizată, regresie, clasificare, arbore de decizie; - caracteristici ale algoritmului de regresie liniară, pentru învățare supervizată; - caracteristici ale algoritmului KNN (k Nearest Neighbours), pentru învățare supervizată; - caracteristici ale algoritmului de clasificare bazat pe arborele de decizie, pentru învățare supervizată; - caracteristici ale rețelelor neuronale, rețele perceptron, principiu de învățare prin rețeaua neuronală.
3. Memorarea datelor și organizarea codului în limbaj de programare	3.1. Comenzi SQL pentru gestionarea unei baze de date - caracteristici și sintaxă ale unor comenzi de bază pentru interogarea bazei de date (SQL): selecție, proiecție, ordonare, operatori, funcții scalare uzuale (de prelucrare a numerelor, a datelor calendaristice, a șirurilor de caractere, conversii), funcții agregate uzuale (pentru sumă, medie, numărare, minim, maxim, de tratare a valorii NULL), grupare, preluare a datelor din mai multe tabele (join), subinterogări; - caracteristici și sintaxă ale unor comenzi de bază pentru manipularea datelor (DML): inserare, modificare, ștergere; - caracteristici și sintaxă ale unor comenzi de bază pentru manipularea structurii bazei de date (DDL): creare a unei tabele, modificare a structurii unei tabele (adăugare a unui câmp, ștergere a unui câmp, modificare a unui câmp), ștergere a unei tabele, secvențe, indecși, vederi (view); - caracteristici și sintaxă ale unor comenzi de bază pentru gestionarea drepturilor de acces la baza de date (DCL): drepturi, acordare și retragere a unor drepturi; - caracteristici și sintaxă ale unor comenzi de bază pentru gestionarea tranzacțiilor din baza de date (TCL): inițiere, puncte de revenire, revenire, finalizare; - repere pentru executarea comenzilor;

Domenii de conținut	Conținuturi
	- repere pentru identificarea și utilizarea altor comenzi, adecvate pentru prelucrarea bazelor de date, în funcție de nevoile proprii. 3.2. Clase ale limbajului Python pentru interacțiunea cu bazele de date - caracteristici; - metode ale claselor principale (de exemplu, Connection, Cursor) pentru operații uzuale: conectare la baza de date, executare a unei comenzi, închidere a conexiunii cu baza de date; - repere pentru identificarea și utilizarea altor clase și metode adecvate pentru interacțiunea cu bazele de date, în funcție de nevoile proprii. 3.3. Biblioteci ale limbajului Python pentru învățare automată - caracteristici, clase și metode specifice din biblioteca Matplotlib pentru vizualizarea datelor: reprezentarea grafică a datelor prin puncte, grafic liniar, diagramă de dispersie, diagramă cu bare, histogramă, diagramă box-plot; - caracteristici, clase și metode specifice din biblioteca Pandas pentru curățarea, manipularea și analiza statistică a datelor: tabele (DataFrames), eliminarea valorilor extreme, obținere a valorilor pentru un atribut, conversii, statistici descriptive (medie, deviație standard, minim, maxim), calcule statistice de bază, determinarea corelațiilor între variabile, agregarea datelor pe categorii, frecvența valorilor unice; - caracteristici, clase și metode specifice din biblioteca Scikit-learn pentru rezolvarea problemelor de codificare și normalizare a datelor, clasificare, regresie și clusterizare: metode pentru antrenare, predicție și evaluare în rezolvarea problemelor de învățare automată; - caracteristici, clase și metode specifice din biblioteca NumPy pentru algebră liniară: operații cu liste de valori, operații cu matrici, rezolvarea sistemelor de ecuații; - repere pentru utilizarea algoritmilor prin metodele disponibile în biblioteci, pentru rezolvarea unor probleme de natură practică; - repere pentru identificarea și utilizarea altor clase și metode adecvate pentru învățare automată, 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 măsoară? 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.

Un context de învățare poate viza simultan mai multe competențe, mai ales atunci când acestea sunt interconectate. De exemplu, pentru rezolvarea unei probleme cu ajutorul calculatorului se vizează, în cadrul aceleiași activități de învățare, atât competențe de nivel cognitiv scăzut (identificarea principalelor caracteristici ale modelelor conceptuale și operaționale implicate, explicarea principiilor care stau la baza acestora, în contextul particular al problemei, utilizarea lor în rezolvare) dar și competențe de nivel cognitiv ridicat (analizarea caracteristicilor modelelor conceptuale și operaționale specifice problemei, evaluarea corectitudinii și eficienței soluțiilor propuse și elaborarea algoritmului personalizat).

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, algoritimizarea, 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 metode/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).

Pentru specializarea matematică-informatică, la clasele cu predarea disciplinei informatică în regim intensiv, **limbajele de programare pentru implementare sunt Python și C++**, conform programei, de exemplu în una dintre variantele:

- în paralel (alocând pentru practică anumite ore, săptămânal, fiecărui limbaj, ca în cazul limbilor străine, sau exemplificând alternativ implementări în cele două limbaje), evidențiind pe parcurs asemănările și deosebirile dintre acestea;
- succesiv, realizându-se o recapitulare a unor concepte, în celălalt limbaj, conform programei școlare.

Predarea elementelor de limbaj specifice Python și C++ în paralel oferă o abordare solidă, pe baza unui plan didactic eficient, având în vedere consolidarea conceptelor comune și susținerea învățării prin comparație (observând asemănări de logică și structură, diferențe de sintaxă, o înțelegere mai profundă a conceptelor de bază, cum ar fi gestionarea automată sau manuală a memoriei), evidențierea complementarității conceptuale (Python oferă o sintaxă simplă, ușor de citit, care permite elevilor să se concentreze pe concepte fără să fie pus accent pe detalii tehnice, iar C++ facilitează înțelegerea profundă a mecanismelor din spatele limbajului — tipuri de date, gestionarea memoriei, pointeri, compilare), formarea unei gândiri flexibile în programare, evidențiindu-se faptul că limbajul este doar un instrument, iar gândirea logică este universală, facilitându-se astfel adaptabilitatea la orice alt limbaj viitor.

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 XII-a, specializarea matematică-informatică, pentru clase cu predarea disciplinei informatică în regim intensiv:

1. *Organizarea conceptuală a datelor. Model conceptual relațional – entitate-relație*
2. *Strategii de rezolvare a problemelor. Forme normale pentru proiectarea modelului conceptual al unei probleme de gestiune*
3. *Memorarea datelor și organizarea codului în limbaj de programare. Clase ale limbajului Python pentru interacțiunea cu bazele de date*
4. *Memorarea datelor și organizarea codului în limbaj de programare. Comenzi SQL pentru gestionarea unei baze de date*
5. *Organizarea conceptuală a datelor. Modele conceptuale de organizare a datelor folosite în învățarea automată*
6. *Strategii de rezolvare a problemelor. Algoritmi utilizați în învățare automată*
7. *Memorarea datelor și organizarea codului în limbaj de programare. Biblioteci ale limbajului Python pentru învățare automată*

Se recomandă introducerea în tematica modelului entitate-relație prin exemple din viața reală (de exemplu, bibliotecă, magazin, școală) prin conectarea permanentă a conceptelor teoretice cu scenarii concrete, astfel încât elevii să înțeleagă conceptele specifice ca „entitate”, „relație”, „identificator unic”.

Procesul de învățare trebuie să fie centrat pe investigare și rezolvare de probleme, iar activitățile să favorizeze explorarea comparativă a soluțiilor. De exemplu, se poate analiza modul în care aceeași problemă de gestiune a datelor poate fi rezolvată atât prin comenzi SQL clasice, cât și prin instrumente moderne de analiză vizuală a datelor (de exemplu, Google Data Studio, Power BI).

Activitățile pot fi centrate pe proiecte de durată (individuale sau de grup) care implică proiectarea și gestionarea bazelor de date, dezvoltarea unor aplicații digitale, astfel încât elevii să poată experimenta întregul ciclu de dezvoltare al unei aplicații informatice – de la analiză și proiectare, până la testare și prezentarea rezultatelor.

Pentru clasa a XII-a, se recomandă ca elevii să valorifice competențele formate pe parcursul claselor anterioare în cadrul unor proiecte complexe, interdisciplinare, axate pe rezolvarea de probleme reale și pe construirea de produse informatice funcționale, integrând instrumente de lucru colaborativ online.

Se recomandă realizarea unui proiect integrator, care ar putea implica realizarea unei baze de date pentru un scenariu dat, în care să se integreze, pe parcurs, elemente din ce în ce mai complexe, pe baza studiului conținuturilor specifice, asigurând documentarea, identificarea necesarului afacerii, realizarea modelului conceptual (cu elemente cât mai variate), maparea acestuia, implementarea unor module ale acestuia, precum și testarea și realizarea documentației aferente. O astfel de integrare ar sublinia coerența practică a conținuturilor predate și ar antrena competențele transversale.

Se recomandă o abordare activă, practică și exploratorie, în care profesorul are rolul de mentor, iar elevul devine creator de soluții, astfel încât procesul de învățare să cultive autonomia, gândirea critică și etica în utilizarea tehnologiei. Profesorul ghidează procesul de învățare prin întrebări, studiu de caz și prin sprijin acordat în momentele-cheie, facilitând trecerea de la învățarea asistată la învățarea autonomă și reflexivă.

Se recomandă integrarea noțiunilor introductive de inteligență artificială în proiecte interdisciplinare. Elevii pot explora modul în care o bază de date alimentată cu exemple reale poate susține un model de învățare automată, pot reflecta critic asupra modului în care AI influențează deciziile, discutând despre biasul datelor, confidențialitate și responsabilitate în contextul discuțiilor despre algoritmi utilizați în învățarea automată.

Se recomandă ca, la finalul clasei a XII-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 crește motivația elevilor și 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;
- UiPath Foundation (<https://aigeneration.uipathfoundation.com/ro/>), platformă ce conține planuri de lecție, prezentări, fișe, teste privind introducerea în inteligența artificială;
- pyML.ro (<https://www.pyml.ro/>), platformă cu lecții pentru introducere în învățare automată.

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,

Nume și prenume	Grad didactic/Titlu științific	Instituție de apartenență, localitate, județ
		Facultatea de Automatică și Calculatoare
NIȚU Alina	profesor, gradul didactic I	Liceul Teoretic „Ovidius”, Constanța, județul Constanța
NURLA Aidan	profesor, gradul didactic I	Colegiul Național Militar „Alexandru Ioan Cuza”, Constanța, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forțelor 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 Național Militar „Dimitrie Cantemir”, Breaza, județul Prahova
TEGLAȘ Maria	profesor, gradul didactic II	Colegiul Național 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
ȚĂPUȘ Nicolae	profesor universitar, doctor	Academia Română
BORIGA Radu Eugen	conferențiar universitar, doctor	Universitatea București, Facultatea de Matematică și Informatică

EDITOR: PARLAMENTUL ROMÂNIEI — CAMERA DEPUTAȚILOR



„Monitorul Oficial” R.A., Str. Parcului nr. 65, sectorul 1, București; 012329
C.I.F. RO427282, IBAN: RO55RNCB0082006711100001 BCR
și IBAN: RO12TREZ7005069XXX000531 DTCPMB (alocat numai persoanelor juridice bugetare)
Tel. 021.318.51.29/150, fax 021.318.51.15, e-mail: marketing@ramo.ro, www.monitoruloficial.ro

Relații cu publicul: șos. Panduri nr. 1, bloc P33, sectorul 5, București; 050651.
Tel. 021.401.00.73, 021.401.00.78/79/83.

Pentru publicări, încărcări actele pe site, la: <https://www.monitoruloficial.ro>, secțiunea Publicări.

