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. 51
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 Python ca limbaj de programare de bază și a limbajelor C++, respectiv a SQL, conform programei școlare corespunzătoare profilului, clasei și specializării;
- utilizarea unor resurse digitale moderne în laboratoare de informatică dotate adecvat;
- corelarea activităților de învățare cu domenii STEM, economie, științe sociale și artă digitală.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

COMPETENȚE GENERALE (CG)

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

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

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

Clasa a 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 înlănțuite (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 înlănțuite, 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* - *identificarea metodelor de reprezentare a arborilor ordonați, cu referințe ascendente sau referințe descendente, pentru exemplificarea reprezentării relațiilor de rudenie dintre strămoși și a descendenților unei familii sub forma unui arbore genealogic* - *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 derivării unei clase, punând în evidență „specializarea” acesteia, de exemplu pentru clasa „Vehicul” derivată în clasele „Autoturism” și „Camion”*
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 explorării unei rețele de comunicații, unde algoritmul explorează în profunzime traseele posibile de transmitere a unor date* - *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, inordine, 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, inordine – consultarea pe niveluri, iar postordine – raportarea rezultatelor de jos în sus* - *precizarea etapelor operațiilor specifice arborelui binar de căutare, în contextul inserării și căutării unor dispozitive dintr-o rețea de comunicații organizată ca arbore binar de căutare, în care fiecare dispozitiv este identificat printr-un cod unic*

Clasa a XI-a
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 utilizat în activitățile unui club de robotică, unde se explorează toate opțiunile posibile, revenindu-se la starea anterioară în momentul în care pentru starea curentă au fost explorate toate variantele* - *identificarea caracteristicilor metodei Backtracking, în contextul utilizării unei liste pentru memorarea obiectivelor turistice vizitate succesiv într-un traseu turistic, cu posibilitatea de revenire la un obiectiv vizitat imediat înaintea celui curent, pentru construirea unei alternative a traseului* - *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 list 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 nivelurilor de acces la membrii 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 constructorilor 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

Clasa a XI-a
- *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* - *explicarea diferenței dintre listele simplu înlănțuite, respectiv listele dublu înlănțuite, 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, 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 numărul de inventar) pentru organizarea ierarhică a informațiilor într-un sistem de comunicații, unde fiecare cheie are o poziție determinată în funcție de valoarea sa* - *explicarea încapsulării, prin exemplul stabilirii membrilor unei clase, „Rețea”, unde datele tehnice nu sunt accesibile pentru utilizator, pentru a reflecta controlul asupra datelor sensibile* - *explicarea derivării unei clase, punând în evidență „specializarea” acesteia, de exemplu pentru clasa „Vehicul” derivată în clasele „Autoturism” și „Camion”*
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 ordinea vizitării nodurilor (depozitelor) și 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 pas cu pas conexiunile cu costul cel mai mic, eliminându-se astfel un număr maxim de conexiuni inutile* - *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: verificare (inordine), planificare (preordine) sau evaluare (postordine)* - *explicarea modului în care inserarea într-un heap menține proprietatea specifică, prin cernere, prin exemplul adăugării unei noi solicitări într-un sistem de prioritizare, 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 list 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 actualiza starea unui dispozitiv 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*

Clasa a XI-a
- *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 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 derivării unei clase, punând în evidență „specializarea” acesteia, de exemplu pentru clasa „Vehicul” derivată în clasele „Autoturism” și „Camion”*
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 acoperirea 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 acoperirea 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 (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* - *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, pentru deblocarea unui dispozitiv, știind că utilizatorul își amintește că ultimele două cifre sunt pare* - *aplicarea etapelor metodei Backtracking pentru generarea tuturor combinărilor 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 list 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” utilizează membri 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”, fiecare având metode specifice, dar bazate pe structura comună de date*

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 (parcurgere înainte) și verificarea inversă a traseului la întoarcere (parcurgere 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 unei mașini care curăță străzile unei localități, pentru a verifica dacă mașina poate parcurge toate străzile* - *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 derivare a unei clase, punând în evidență „specializarea” acesteia, de exemplu pentru clasa „Vehicul” derivată în clasele „Autoturism” și „Camion”*
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 vulnerabilitățile în lanțul de transmitere a datelor* - *analizarea metodelor BFS și DFS de parcurgere a unui graf neorientat/orientat în vederea adaptării lor pentru determinarea 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 toate 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
- *analizarea 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* - *analizarea 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* - *analizarea 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
- *analizarea 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* - *analizarea aplicabilității pointerilor în C++ în contextul examinării unui fragment de cod pentru identificarea potențialelor erori* - *analizarea aplicabilității pointerilor în implementarea unei funcții C++ care primește ca argument un pointer la o variabilă întreagă și modifică valoarea variabilei prin intermediul pointerului* - *analizarea 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* - *analizarea 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* - *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* - *analizarea instrumentelor 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* - *analizarea instrumentelor clasei list 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* - *analizarea 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
- *analizarea avantajelor și dezavantajelor utilizării membrilor privați și publici 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)* - *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* - *analizarea 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)* - *analizarea 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
- *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*

Clasa a XI-a
- *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 derivării unei clase, punând în evidență „specializarea” acesteia, de exemplu pentru clasa „Vehicul” derivată în clasele „Autoturism” și „Camion”*
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 rutei 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 echilibru în procesul decizional* - *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 bază de date ierarhizată 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* - *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*

Clasa a XI-a
- *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, cu precizarea destinației* - *argumentarea utilizării instrumentelor clasei list 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 nivelurilor de acces pentru garantarea protecției 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 private și 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 utilizează date ale unor clase definite anterior, de exemplu, clasa „Optiune” utilizează membri din clasele „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

Clasa a XI-a
- *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 clasa „Vehicul” derivată în clasele „Autoturism” și „Camion”*
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 între fiecare 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, 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 pentru un sistem de recomandare a ordinii de parcurgere a bibliografiei pentru studiul unei discipline pentru elevii de liceu, parcurgerea unei lucrări fiind condiționată de parcurgerea anterioară a altora* - *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 acoperirii 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, atribuiri, 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 măsor? Ce feedback dau? Ce decizii iau?;
- diferențiere/personalizare: adaptarea parcursului didactic la situații specifice (de exemplu, elevi cu CES și/ sau dizabilități, elevi cu ritm înalt de învățare, elevi care au nevoie de învățare remedială, elevi în risc de abandon etc.).

Orientările metodologice generale cuprind:

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

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

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

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

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

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

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

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

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

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

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

Pentru specializarea matematică-informatică, la clasele cu predarea disciplinei informatică în regim intensiv, **limbajul de programare de bază pentru implementare este Python**, secondat de limbajul 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 limbajul C++, 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.
Predarea succesivă presupune ca elevii să utilizeze mai întâi limbajul Python, să stăpânească bazele, iar abia apoi să treacă la C++, pentru a aprofunda sau extinde conceptele deja învățate. Începând cu Python, elevii se concentrează pe logică, algoritmi și gândire procedurală, fără a pune accentul pe detalii tehnice, iar trecerea la C++ se face cu o bază solidă: ei știu deja să elaboreze algoritmi și să îi reprezinte în limbaj de programare, și învață doar cum se implementează aceștia în alt limbaj de programare, precum și aspecte noi, specifice acestui limbaj.
În exemplele de activități de învățare, limbajul de programare vizat este limbajul de bază studiat, Python, cu excepția cazului în care se precizează explicit limbajul C++.

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, combinări, 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 exemplificarea 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 înlănțuite în Python se face doar prin câteva exemple, utilizând clasa list, fiecare înlănțuire 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 înlănțuire 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 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 inteligenţă computaţională care facilitează învăţarea 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 şi 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 Naţional Militar „Alexandru Ioan Cuza”, Constanţa, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forţelor Terestre „Nicolae Bălcescu”, Sibiu, județul Sibiu
PETRESCU Manuela-Andreea	lector universitar, doctor	Universitatea Babeș-Bolyai, Cluj-Napoca, județul Cluj
PINTEA Adrian-Doru	profesor, gradul didactic I	Colegiul Național „Andrei Mureșanu”, Dej, județul Cluj
PINTEA Rodica	profesor, gradul didactic I	Liceul Teoretic „Radu Vlădescu”, Pătârlagele, județul Buzău
POP Florin	profesor universitar, doctor	Universitatea Națională de Științe și Tehnologie Politehnica București, Facultatea de Automatică și Calculatoare
POSTOLACHE Florin	lector universitar, doctor	Academia Navală „Mircea cel Bătrân” Constanța, Facultatea de Inginerie Marină, județul Constanța
SĂCUIU Silviu Eugen	profesor, gradul didactic I	Colegiul Național „Mihai Viteazul”, București
SPĂTARU Adrian	lector universitar, doctor	Universitatea de Vest, Timișoara, Facultatea de Matematică și Informatică, județul Timiș
STANCIU Diana	profesor, gradul didactic I	Colegiul Naţional Militar „Dimitrie Cantemir”, Breaza, județul Prahova
TEGLAȘ Maria	profesor, gradul didactic II	Colegiul Naţional Militar „Mihai Viteazul”, Alba Iulia, județul Alba
TÎMPLARU Roxana-Gabriela	profesor, gradul didactic I	Colegiul „Ștefan Obobleja”, Craiova, județul Dolj
UNGUREANU Florentina	profesor, gradul didactic I	Colegiul Național de Informatică, Piatra Neamț, județul Neamț
VINȚ Corina Elena	profesor, gradul didactic I	Colegiul Național de Informatică „Tudor Vianu”, București

COORDONATORI/RESPONSABILI/CONSULTANȚI ȘTIINȚIFICI

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