Programa școlară
pentru disciplina

Informatică

Clasa a X-a

Curriculum de specialitate (CS)

Filiera vocațională, profilul militar,
specializarea matematică-informatică militară

Învățământ liceal

- 2026 –

Anexa nr. 45
Programa școlară pentru disciplina *Informatică*, clasa a X-a, curriculum de specialitate (CS)

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

NOTĂ DE PREZENTARE

Statutul disciplinei

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

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

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

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

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

Activitățile practice sunt desfășurate obligatoriu în laboratorul de informatică.

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

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

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

Rolul disciplinei în formarea elevilor

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

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

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

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

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

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

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

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

Categorii de programe școlare pentru disciplina *informatică*

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

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

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

Orientări generale și specifice în lectura programei școlare

Aplicarea programei presupune:

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

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

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

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

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

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

Setul de competențe generale ale disciplinei *informatică* este construit pe baza taxonomiei Bloom (revizuite), urmărind o dezvoltare progresivă a gândirii logice și algoritmice, de la nivelurile de înțelegere și aplicare până la cele de analiză, evaluare și

creare. Această structură sprijină elevii în formarea unei viziuni integrate asupra procesului de dezvoltare software, de la concept la implementare, asigurând totodată experiență de învățare în domeniul informaticii.

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

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

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

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

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

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

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

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

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

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

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

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

COMPETENȚE GENERALE (CG)

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

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

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

Clasa a X-a
CS 1.1. Identifică principalele caracteristici ale organizării datelor în cadrul unor modele conceptuale simple - neliniare, liniare, asociative, mixte, pentru a structura și accesa date în vederea prelucrării acestora
- *definirea conceptului de mulțime în contextul determinării drepturilor distincte de acces care sunt atribuite unui grup de utilizatori cu roluri diferite, unde fiecare drept apare o singură dată, chiar dacă e atribuit mai multor persoane, iar ordinea înregistrării acestora nu contează* - *enumerarea operațiilor de bază asupra mulțimilor (reuniune, intersecție, diferență, egalitate, incluziune, testare apartenență), în contextul în care se cunoaște lista obiectelor transportate cu două rucsacuri și se cere să fie identificate obiectele comune, cele conținute de un singur rucsac etc.* - *precizarea caracteristicilor unei mulțimi în contextul gestionării utilizatorilor înregistrați pe platforme diferite pentru a evita dublurile* - *definirea conceptului de șir de caractere ca o colecție ordonată și imutabilă de caractere, prin analogie cu textul unui eseu sau cu un mesaj scris pe telefon* - *enumerarea operațiilor de bază asupra șirurilor de caractere (acces la un caracter sau la o secvență de caractere, concatenare, comparare lexicografică, căutare) în contextul verificării sau construirii unui nou mesaj pe baza unui text trimis de un elev către colegii săi pentru organizarea unei activități școlare* - *definirea conceptului de structură asociativă (dicționar) cu evidențierea faptului că se rețin perechi cheie-valoare și se permite accesul rapid la date prin intermediul cheilor, de exemplu, într-o aplicație de traducere, în care fiecare cuvânt este o „cheie", iar traducerea sa este „valoarea" sau în contextul asocierii dicționarului cu agenda telefonică în care fiecărui număr de telefon (cheie) îi corespunde un nume (valoare)* - *enumerarea cheilor și a valorilor asociate dintr-un dicționar pentru un formular online, în care fiecare câmp completat (de exemplu, „nume", „vârstă", „email") este asociat cu valorile corespunzătoare* - *identificarea caracteristicilor unui model mixt, de tipul listă de liste, pentru memorarea mediilor elevilor dintr-o clasă, în care fiecare listă corespunde câte unui elev și conține mediile pentru fiecare disciplină, astfel încât elementul aflat în lista i, pe poziția j, reprezintă media elevului i la disciplina j;* - *identificarea caracteristicilor unui model mixt, de tipul dicționar de liste și a modului de organizare a datelor în contextul evidenței școlare a claselor, în care fiecare clasă are o listă de elevi* - *enumerarea metodelor de acces și parcurgere a datelor într-o listă de dicționare în contextul gestiunii unor comenzi, fiecare comandă fiind reprezentată printr-un dicționar (prin asocierea produs-cantitate) pentru a genera facturile corespunzătoare* - *enumerarea caracteristicilor tuplurilor în contextul utilizării lor pentru stocarea coordonatelor geografice (latitudine, longitudine) într-un sistem de navigație care necesită date ce nu trebuie să fie modificate accidental* - *identificarea caracteristicii tuplurilor de reprezentare a unor perechi fixe de valori în contextul memorării culorilor ca triplete RGB, într-o aplicație de grafică* - *enumerarea caracteristicilor modelelor conceptuale liniare tuplu și listă, în contextul reprezentării unui punct fix de pe o hartă (de exemplu, un oraș, ale cărui coordonate nu pot fi modificate) și a unui punct mobil de pe hartă (care se actualizează, de exemplu, poziția unui robot care se mișcă)*
CS 1.2. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea listelor, a șirurilor de caractere
- *recunoașterea caracteristicilor metodei căutării binare cu specificarea condiției necesare, în contextul localizării rapide a unui contact într-o agendă telefonică ordonată alfabetic* - *enumerarea etapelor metodei căutării binare în contextul localizării unui titlu într-o listă ordonată de cărți dintr-o bibliotecă digitală* - *recunoașterea caracteristicilor metodei interclasării în contextul combinării a două liste ordonate de produse provenite din depozite diferite pentru a obține o singură listă finală, ordonată* - *enumerarea pașilor necesari aplicării metodei interclasării în contextul unificării a două liste ordonate de participanți înscriși online și offline la un eveniment, astfel încât rezultatul să fie o listă unică și ordonată* - *enumerarea unor metode simple de criptare (de exemplu, cifrul lui Cezar, substituția monoalfabetică) în contextul utilizării unor tehnici de codificare pentru a ascunde indicii într-un joc de tip „vânătoare de comori"*

Clasa a X-a
- *identificarea caracteristicilor specifice fiecărei metode de criptare (de exemplu, cifrul lui Cezar - deplasarea literelor, substituție monoalfabetică - înlocuire literă cu literă) în contextul analizării unui mesaj codificat, observând dacă literele sunt deplasate, înlocuite sau reordonate*
CS 1.3. Identifică specificul, caracteristicile și etapele unor strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor
- *definirea caracteristicilor metodei Divide et impera în contextul unei discuții ghidate în care sunt reliefate avantajele metodei prin analogie cu fragmentarea unei activități complexe în subactivități realizate de echipe diferite, urmate de reunirea rezultatelor* - *enumerarea pașilor specifici metodei Divide et impera (divizarea problemei, rezolvarea subproblemelor, combinarea soluțiilor) în contextul aplicării acestei strategii la algoritmul de determinare a valorii maxime dintr-un șir de numere memorate* - *definirea caracteristicilor metodei Greedy în contextul alegerii celei mai bune opțiuni la fiecare pas, atunci când un elev își planifică trei activități dintr-o listă în care sunt asociate duratele activităților, astfel încât să maximizeze timpul liber* - *recunoașterea caracteristicilor algoritmilor care utilizează metoda Greedy, în contexte aplicative, precum selectarea celor mai avantajoase oferte într-o aplicație de cumpărături, în care deciziile sunt luate secvențial, fără posibilitatea revenirii asupra alegerilor anterioare*
CS 1.4. Identifică principalele elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele simple - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice
- *recunoașterea sintaxei corecte de inițializare a unei mulțimi (set) în contextul în care sunt adunate obiecte diferite (de exemplu, tipuri de fructe) într-o colecție în care elementele sunt unice, cu evidențierea modului în care utilizarea acoladelor și a elementelor separate prin virgule permite crearea unei mulțimi fără dubluri și fără ordine prestabilită* - *identificarea operatorului pentru verificarea apartenenței unui element la o mulțime în Python în contextul scrierii unei secvențe de instrucțiuni prin care se verifică dacă o persoană este deja înscrisă pe lista participanților la un eveniment, fără a fi necesară parcurgerea întregii liste* - *identificarea modalităților de reprezentare a șirurilor de caractere în Python (apostrofuri simple, triple și ghilimele) prin asocierea cu modurile în care putem cita sau nota o expresie în vorbirea curentă: între apostrofuri, ghilimele simple sau ghilimele/apostrofuri triple în contexte cotidiene ca notarea unui titlu, a unui citat sau a unui text mai lung (paragraf, textul unui e-mail etc.)* - *recunoașterea unei modalități de accesare a unui caracter dintr-un șir în contextul utilizării unei litere dintr-un cuvânt, la o anumită poziție cu evidențierea faptului că, în Python, caracterele dintr-un șir pot fi accesate prin indexare, așa cum putem număra pozițiile literelor dintr-un cuvânt* - *identificarea modului de accesare a unui caracter dintr-un șir de caractere folosind operatorul de indexare, în contextul afișării primei litere dintr-un nume introdus într-un formular digital* - *descrierea corespondenței dintre caracterele unui șir și valorile lor ASCII în contextul codificării unui text prin înlocuirea fiecărui caracter cu codul său ASCII reprezentat prin trei cifre* - *recunoașterea notării specifice cu acolade și a separatorului între cheie și valoare în cazul dicționarelor, prin analogie cu organizarea informațiilor într-o listă de contacte: numele persoanei urmat de simbolul „:”, apoi de numărul de telefon, elementele fiind grupate între acolade* - *enumerarea operațiilor de bază asupra dicționarelor (creare, adăugare, ștergere, actualizare) prin analogie cu actualizarea listei de contacte din agenda telefonică: crearea unei liste de contacte, adăugarea unui contact nou, ștergerea unui contact existent, modificarea numărului de telefon al unei persoane* - *identificarea elementelor de sintaxă ale limbajului Python pentru inițializarea unei variabile corespunzătoare memorării datelor organizate sub forma unui model mixt, de tipul listă de liste, pentru memorarea relațiilor de vecinătate în cadrul unui grup de țări, în care fiecare listă corespunde câte unei țări, astfel încât elementul aflat în lista i, pe poziția j, este 1, dacă țările i și j sunt vecine și este 0, altfel* - *identificarea elementelor de sintaxă ale limbajului C++ pentru inițializarea unei variabile de tip tablou bidimensional, corespunzătoare memorării datelor organizate sub forma unui model mixt, de tipul listă de liste, pentru memorarea relațiilor de prietenie în cadrul unui grup de persoane, în care fiecare listă corespunde câte unei persoane, astfel încât elementul aflat în lista i, pe poziția j, este 1, dacă persoanele i și j sunt prietene și este 0, altfel* - *identificarea modului de inițializare a unui tuplu în Python, pentru un punct fix pe o hartă, dat prin coordonatele (x,y)* - *recunoașterea sintaxei de accesare a componentelor unui tuplu în Python, în contextul accesării abscisei sau ordonatei unui punct fix de pe o hartă, reprezentat printr-un tuplu*

Clasa a X-a
CS 1.5. Identifică elementele de sintaxă și componentele din definiția și apelul subprogramelor recursive, pentru a le utiliza în implementarea algoritmilor în limbaj de programare
- *enumerarea caracteristicilor principale ale unui subprogram recursiv, în contextul elaborării unui astfel de subprogram pentru calculul lui n! definit prin relația recurentă cunoscută* - *recunoașterea termenilor „recursivitate directă” și „recursivitate indirectă” prin asocierea lor cu definițiile corespunzătoare* - *recunoașterea sintaxei corecte pentru definirea unui subprogram recursiv în Python, în contextul calculării sumei cifrelor unui număr* - *enumerarea condițiilor pe care trebuie să le îndeplinească un subprogram recursiv pentru a funcționa corect, menționând necesitatea unui caz de oprire, a unei transformări care reduce progresiv problema și a unui apel recursiv care aplică aceeași procedură până la atingerea condiției de oprire, în contextul calculării factorialului unui număr*

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

Clasa a X-a
CS 2.1. Explică principiile de organizare și prelucrare a datelor în cadrul unor modele conceptuale simple - neliniare, liniare, asociative, mixte, pentru a le gestiona eficient
- *explicarea caracteristicilor mulțimilor și listelor, în contextul organizării și accesării elementelor acestora în situații cotidiene, de exemplu, o listă de persoane care așteaptă la rând și determinarea clienților distincți ai unei companii de telefonie, pe baza unei mulțimi de abonamente (în condițiile în care un client poate avea mai multe astfel de abonamente)* - *explicarea modului în care mulțimile sunt utilizate pentru clasificarea pe categorii a produselor într-un magazin online, evidențiind cum apartenența la o categorie permite organizarea și filtrarea eficientă a articolelor* - *explicarea modului în care este prelucrat un șir de caractere, în contextul funcționării unui sistem de verificare a complexității parolelor, care analizează fiecare simbol pentru a identifica litere mari, litere mici, cifre și caractere speciale* - *explicarea modului în care concatenarea și separarea șirurilor de caractere sunt utilizate pentru combinarea sau divizarea informațiilor textuale în contextul generării automate a etichetelor pentru colete, unde numele destinatarului și adresa sunt reunite sau separate în funcție de necesitățile sistemului* - *explicarea rolului cheilor unice și a valorilor asociate în dicționare, în contextul serviciului de evidența populației, unde nu pot exista două CNP-uri identice (chei unice), iar modificarea numelui unei persoane actualizează valoarea corespunzătoare cheii respective* - *explicarea modului în care un dicționar poate fi folosit pentru a organiza toate informațiile unei comenzi online, fiecare cheie din dicționar reprezentând un tip de informație (de exemplu, „client”, „adresă”, „produse”), iar valoarea asociată acelei chei conținând detaliile respective, astfel încât toate datele comenzii să fie grupate logic și să fie ușor de găsit* - *explicarea modului de acces la date în cazul unui model mixt, de tipul listă de liste, pentru memorarea mediilor elevilor dintr-o clasă, în care fiecare listă corespunde câte unui elev și conține mediile pentru fiecare disciplină, astfel încât elementul aflat în lista i, pe poziția j, reprezintă media elevului i la disciplina j;* - *explicarea modului în care un dicționar este folosit pentru a structura datele într-un sistem de gestiune a proiectelor, fiecare cheie reprezentând numele unei echipe, iar valoarea asociată fiind lista sarcinilor acelei echipe, ceea ce permite vizualizarea clară și separată a responsabilităților fiecărei echipe* - *explicarea modului în care structura unui orar poate fi reprezentată printr-un dicționar, în care fiecare cheie corespunde unei zile, iar valoarea asociată este lista orelor din acea zi, ceea ce permite accesul rapid la programul complet al unei anumite zile* - *explicarea caracteristicilor unui tuplu în contextul unei aplicații financiare, evidențiind motivul pentru care tuplurile sunt folosite pentru informații fixe (precum perechi de tipul „nume client – CNP”), astfel încât datele care nu trebuie schimbate să rămână protejate* - *explicarea modului în care tuplurile sunt adecvate pentru stocarea înregistrărilor fixe într-o aplicație de calendar, unde fiecare eveniment este reprezentat printr-un tuplu (data, ora, titlu_eveniment), ceea ce oferă o structură constantă și ușor de prelucrat pentru toate intrările*
CS 2.2. Explică etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea listelor, a șirurilor de caractere
- *explicarea etapelor metodei căutării binare în contextul găsirii rapide a unui contact într-o agendă telefonică ordonată alfabetic, evidențiind faptul că numele căutat este localizat prin împărțirea repetată a listei în jumătăți succesive* - *explicarea importanței ordonării elementelor într-o listă, arătând că doar o listă sortată permite utilizarea eficientă a căutării binare (de exemplu, pentru a identifica rapid un zbor într-o listă ordonată după ora plecării) și a interclasării (de exemplu, pentru a combina două liste ordonate de rezervări ale unei săli într-un registru unic, identificând eventualele suprapuneri)*

Clasa a X-a
- *explicarea etapelor metodei de interclasare, arătând cum două liste ordonate sunt combinate într-una singură prin compararea elementelor lor pe rând, folosind ca exemplu situația unui hotel care primește rezervări din două surse diferite (formular online și agenție de turism), fiecare sursă generând propria listă ordonată, iar hotelul având nevoie să le reunească într-o singură listă cu rezervări ordonate cronologic* - *explicarea modului de funcționare al diferitelor metode de criptare (de exemplu, cifrul lui Cezar, substituție simplă, cifrul Vigenère) în contextul protejării unui mesaj trimis online, evidențiind cum transformările aplicate textului urmăresc ascunderea semnificației lui* - *explicarea modului în care diferitele metode de criptare produc modificări specifice asupra unui șir de caractere, în contextul mesajelor codificate din jocuri de tip „vânătoare de comori”, evidențiind faptul că literele pot fi deplasate, înlocuite sau reordonate în funcție de metoda utilizată* - *descrierea modului în care metodele simple de criptare se aplică pe exemple concrete, de exemplu, folosirea cifrului lui Cezar pentru a transforma cuvântul „ANA” în „DQC” prin deplasarea fiecărei litere cu 3 poziții în alfabet, utilizarea unui cifru prin substituție pentru a codifica cuvântul „CARTE” folosind un tabel în care C→M, A→Q, R→T, T→A, E→Z, respectiv aplicarea cifrului Vigenère pentru a cripta un mesaj cu un cuvânt cheie, obținând un șir în care fiecare literă este modificată adunând numărul de ordine din alfabet al literei din mesaj cu numărul de ordine al caracterului din cheia de criptare modulo 26*
CS 2.3. Explică etapele unor strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor
- *explicarea etapelor metodei Divide et impera (descompunere, rezolvare, combinare), folosind ca exemplu calcularea sumei unor numere dintr-o listă, prin împărțirea repetată a acesteia și a fiecărei liste obținute pe parcurs în jumătăți (până când se ajunge la liste formate dintr-un singur număr, pentru care se cunoaște rezultatul) și determinarea rezultatului pentru o astfel de listă, prin adunarea sumelor corespunzătoare celor două jumătăți, după obținerea acestora* - *explicarea modului în care se aplică metoda Divide et impera, descriind clar cele trei etape ale sale — descompunerea problemei în subprobleme mai mici, rezolvarea fiecărei subprobleme, apoi combinarea rezultatelor în exemple reale în care această abordare este folosită, precum procesarea imaginilor sau navigația GPS* - *explicarea principiului metodei Greedy pentru rezolvarea unei situații concrete, de exemplu, selectarea, pentru un muncitor, a unui număr maxim de sarcini dintr-o listă de activități, care trebuie rezolvate într-un timp dat, prin alegerea repetată a sarcinii care consumă cel mai puțin timp disponibil* - *explicarea necesității demonstrării corectitudinii utilizării metodei Greedy comparând situații în care această metodă poate conduce uneori la rezultate greșite, de exemplu, pentru problema formării unei sume de bani date cu minimizarea numărului de monede necesare, atunci când sistemul de monede nu este „canonic” (de exemplu, pentru suma 6 și setul de monede 1, 3, 4 metoda Greedy oferă un rezultat necorespunzător, cu trei monede, adică 4+1+1, în loc de soluția de două monede, adică 3+3)* - *explicarea etapelor metodei Greedy aplicate unei probleme practice de planificare, precum selectarea unui număr maxim de spectacole care se pot desfășura în aceeași sală, într-o zi, fără suprapuneri, cunoscându-se intervalele de desfășurare a spectacolelor*
CS 2.4. Explică rolul principalelor elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele simple - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice
- *explicarea modalității de parcurgere a elementelor unei mulțimi în Python prin analogie cu procesul de verificare a fiecărui obiect dintr-o cutie care conține piese unice, fără a ține cont de ordinea lor, evidențiind modul în care o instrucțiune repetitivă permite accesarea fiecărui element al mulțimii* - *explicarea efectului aplicării metodelor (add(), remove(), union(), intersection()) și operatorilor (|, &, -) asupra a două mulțimi în Python, în contextul ilustrării în limbaj de programare a unor exemple inspirate din viața cotidiană (de exemplu, compararea listelor de prieteni din două rețele sociale, respectiv a prietenilor existenți în oricare rețea, a prietenilor comuni, a prietenilor prezenți doar în una dintre rețele)* - *explicarea efectului aplicării metodelor uzuale ale clasei str, în contextul ilustrării în limbaj de programare a unor exemple inspirate din viața cotidiană, de exemplu, separarea cuvintelor dintr-o propoziție (split()), transformarea textului în litere mici sau mari pentru uniformizare (lower(), upper()), eliminarea spațiilor nedorite de la începutul sau sfârșitul unui mesaj (strip()), înlocuirea unui cuvânt greșit (replace()), numărarea aparițiilor unui caracter într-un text (count()), verificarea faptului că un text conține doar litere (isalpha()) sau doar cifre (isdigit())* - *explicarea efectului aplicării unor metode specifice dicționarelor prin asocierea acestora cu gestionarea informațiilor din agenda telefonică, de exemplu, verificarea faptului că un nume se află în agendă (get()), obținerea listei care conține doar numele din agendă (keys()), obținerea listei care conține doar numerele de telefon din agendă (values()), obținerea listei complete nume-număr din agendă (items())* - *explicarea rezultatelor obținute în urma iterării asupra unui dicționar, cu exemplificare pe agenda telefonică, din care, în urma iterării, se poate obține doar lista de nume, doar lista de numere de telefon sau lista de perechi nume-număr de telefon, în funcție de metoda de parcurgere utilizată* - *explicarea utilizării operatorului de indexare pentru accesarea unui element corespunzător unei variabile Python în care sunt memorate date organizate sub forma unui model mixt, de tipul listă de liste, pentru memorarea relațiilor de vecinătate*

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

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

Clasa a X-a
CS 3.1. Utilizează modul de organizare și prelucrare a datelor în cadrul unor modele conceptuale simple - neliniare, liniare, asociative, mixte, pentru a rezolva probleme de gestionare a datelor
- *aplicarea operațiilor de reuniune și intersecție asupra a două mulțimi de clienți din două magazine online, pentru a stabili atât lista tuturor clienților care pot primi oferte promoționale, cât și clienții comuni celor două magazine* - *utilizarea mulțimilor pentru generarea elementelor unice (de exemplu, coduri de acces fără repetiție), verificarea dublurilor într-o listă de elemente (de exemplu, produse identice într-un inventar), verificarea faptului că o colecție conține toate valorile dintr-o secvență de elemente (de exemplu, verificarea faptului că toate zilele unei săptămâni au fost completate într-un raport)* - *utilizarea comparării lexicografice pentru ordonarea titlurilor de cărți într-o bibliotecă digitală, astfel încât acestea să fie afișate alfabetic într-o listă* - *aplicarea operațiilor specifice de parcurgere a unui șir de caractere pentru a verifica dacă acesta reprezintă un număr de telefon (conține exclusiv cifre și are exact 10 caractere)* - *aplicarea operației de concatenare a șirurilor de caractere pentru a construi un mesaj complet, folosind date separate precum numele unui elev și media acestuia, într-un text personalizat* - *utilizarea organizării datelor sub formă de dicționar pentru realizarea unui glosar online care asociază niște termeni din tehnologia informației (chei) cu explicațiile lor (valori), facilitând învățarea rapidă a vocabularului de specialitate* - *utilizarea dicționarelor pentru organizarea perechilor cheie-valoare și obținerea unor rezultate, în contextul parcurgerii listei de produse dintr-un magazin unde fiecare produs (cheia) are asociat un preț (valoarea) și obținerea de rezultate precum valoarea totală a stocului, prețul minim/maxim, media prețurilor, produsele cu prețul mai mare/mai mic decât o valoare dată etc.*

Clasa a X-a
- *accesarea unui element dintr-un dicționar utilizat într-o aplicație de comerț online, în care codul unic al produsului este folosit drept cheie, iar valorile asociate sunt informațiile despre produs (denumire și preț), pentru a afișa rapid detaliile corespunzătoare codului introdus de utilizator* - *utilizarea unui model mixt, de tipul listă de liste, pentru memorarea mediilor elevilor dintr-o clasă, în care fiecare listă corespunde câte unui elev și conține mediile pentru fiecare disciplină, astfel încât elementul aflat în lista i, pe poziția j, reprezintă media elevului i la disciplina j;* - *utilizarea tuplurilor pentru reprezentarea coordonatelor geografice într-o aplicație de navigaţie GPS, prin memorarea fiecărui punct de interes într-un tuplu format din două valori fixe (latitudine, longitudine), astfel încât aceste perechi să poată fi folosite direct în calcule precum determinarea distanței dintre două locații* - *aplicarea unui algoritm de sortare asupra unei liste de produse reprezentate prin tupluri (denumire, preț), folosind al doilea element al fiecărui tuplu pentru a ordona produsele crescător după preț* - *utilizarea tuplurilor pentru calcularea mediei la o disciplină școlară, prin memorarea notelor unui elev în tupluri de forma (disciplină, notă), selectarea tuplurilor corespunzătoare acelei discipline, adunarea notelor și împărțirea sumei la numărul de note adunate pentru a obține media*
CS 3.2. Aplică algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea listelor, a șirurilor de caractere
- *aplicarea metodei de căutare binară pentru găsirea unui element într-o listă ordonată, folosind ca exemple căutarea unui număr de telefon într-o agendă electronică în care contactele sunt aranjate în ordine crescătoare după numărul de telefon* - *aplicarea metodei de căutare binară pentru o listă dată de programări dintr-o zi, ordonate cronologic, pentru a verifica dacă se poate face o programare pentru ora „11:00”, condiția fiind ca la acea oră să nu existe deja o programare* - *aplicarea metodei de căutare binară prin urmărirea pas cu pas a valorilor limitelor de căutare și a poziției elementului verificat, pentru un set dat de valori ordonate, până la identificarea elementului căutat sau stabilirea faptului că acesta nu există în listă, în contextul unei liste ordonate de coduri ale unor produse* - *utilizarea metodei interclasării pentru combinarea a două liste date de produse provenind din două depozite, produsele din fiecare listă fiind ordonate crescător în funcție de preț, obținându-se lista finală cu toate produsele ordonată crescător în funcție de preț* - *aplicarea algoritmilor de criptare corespunzători metodelor studiate pentru transformarea unui mesaj obișnuit într-un mesaj criptat, folosind reguli precise de modificare a caracterelor, în scopul protejării informațiilor transmise* - *exersarea unui sistem de criptare a mesajelor folosind cifrul lui Cezar, prin introducerea unui text format din litere mari și a unei chei de criptare care indică numărul de poziții cu care fiecare literă este deplasată, obținând mesajul criptat* - *utilizarea metodei substituției simple pentru transmiterea de mesaje confidențiale între elevii unei clase, prin înlocuirea fiecărei litere din mesaj cu o altă literă stabilită conform unui tabel de substituție cunoscut, astfel încât mesajul rezultat să poată fi citit doar de destinatarii care cunosc regula de corespondență*
CS 3.3. Aplică strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor
- *aplicarea metodei Divide et impera pentru determinarea sumei necesare pentru plata unor produse, dându-se o listă care conține prețurile lor, prin împărțirea repetată a acesteia și a fiecărei liste obținute pe parcurs în jumătăți (până când se ajunge la liste formate dintr-o singură valoare, pentru care se cunoaște rezultatul) și determinarea rezultatului pentru o astfel de listă, prin adunarea sumelor corespunzătoare celor două jumătăți, după obținerea acestora* - *aplicarea metodei Divide et impera în rezolvarea unor probleme clasice, precum mutarea obiectelor între suporturi respectând reguli precise (problema turnurilor din Hanoi), determinarea valorii maxime sau minime dintr-o listă de date sau localizarea rapidă a unei informații într-o evidență ordonată (căutarea binară)* - *aplicarea metodei de sortare prin interclasare pentru obținerea unei liste ordonate pornind de la o listă neordonată formată din 7 numere date* - *aplicarea metodei Greedy pentru rezolvarea unei situații concrete, de exemplu, selectarea pentru o anumită zi și un timp dat, a unui număr maxim de sarcini dintr-o listă de activități, alegând de fiecare dată sarcina care consumă cel mai puțin timp* - *aplicarea metodei Greedy pentru rezolvarea unei probleme practice, precum selectarea dintr-o listă de spectacole candidate a se desfășura într-o sală, a unui număr maxim de spectacole ce pot fi programate fără a exista suprapuneri*
CS 3.4. Utilizează principalele elemente ale limbajului de programare pentru prelucrarea datelor organizate în modele simple - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice
- *utilizarea unor operatori (|, &, -) și metode (add(), remove(), union(), intersection()) ale clasei set în contexte practice, de exemplu, pentru gestionarea participării elevilor la activități extrașcolare, cu evidențierea modului în care aceste operații pot simplifica prelucrarea și compararea datelor* - *utilizarea unor metode adecvate din clasa set în implementarea unor programe pentru verificarea faptului că o listă este formată din elemente distincte (de exemplu, coduri unice de participare), eliminarea elementelor duplicate dintr-o listă (de exemplu, coduri introduse de mai multe ori), determinarea elementelor comune între două colecții (de exemplu, elevii înscriși la două cercuri diferite)*

Clasa a X-a
- *utilizarea metodelor clasei str în diferite situații de procesare a textului în contexte cotidiene cum ar fi formatarea automată a numelor participanților la un concurs după un model dat (lower(), upper(), title(), capitalize()), validarea unui nume prin verificarea faptului că textul introdus conține doar litere (isalpha()), verificarea dacă un cod introdus conține doar cifre (isdigit())* - *utilizarea metodelor clasei str în diferite situații de modificare a textului în contexte cotidiene cum ar fi curățarea datelor introduse de utilizatori într-un formular online prin eliminarea spațiilor suplimentare (strip()), separarea cuvintelor dintr-o propoziție pentru o prelucrare ulterioară a acestora (split())* - *aplicarea metodelor uzuale ale clasei dict în contexte practice, de exemplu, pentru gestionarea produselor dintr-un magazin: actualizarea stocului de produse (update()), ștergerea unui articol din magazin (pop()), ștergerea ultimului produs adăugat (popitem()), lichidare de stoc (clear()),crearea unei copii de siguranță a stocului înainte de modificări majore (copy()), crearea unui nou stoc pornind de la un stoc existent (fromkeys()) în care vor fi toate produsele inițiale (cheile) dar prețurile aferente (valorile) au valoarea None, verificarea existenței unui produs în magazin cu scopul obținerii prețului acestuia și evitarea erorilor în cazul în care produsul lipsește (get()), adăugarea automată a unui produs în magazin dacă nu există deja (setdefault()), obținerea listelor cu denumirile produselor pentru inventar (keys()), prețurile din magazin (values()) sau perechile produs-preț (items()) pentru afișare pe etichete* - *utilizarea de dicționare imbricate, cu exemplificare pe gestionarea stocului unui magazin (reprezentat sub formă de dicționar) în care fiecare produs (cheia) are asociat un alt dicționar cu detalii ca preț, cantitate etc.* - *utilizarea operatorului de indexare pentru accesarea unui element corespunzător unei variabile Python în care sunt memorate date organizate sub forma unui model mixt, de tipul listă de liste, pentru memorarea relațiilor de vecinătate în cadrul unui grup de țări, în care fiecare listă corespunde câte unei țări, astfel încât elementul aflat în lista i, pe poziția j, este 1, dacă țările i și j sunt vecine și este 0, altfel* - *utilizarea operatorului de indexare pentru accesarea unui element corespunzător unei variabile C++ de tip tablou bidimensional, în care sunt memorate date organizate sub forma unui model mixt, de tipul listă de liste, pentru memorarea relațiilor de prietenie în cadrul unui grup de persoane, în care fiecare listă corespunde câte unei persoane, astfel încât elementul aflat în lista i, pe poziția j, este 1, dacă persoanele i și j sunt prietene și este 0, altfel* - *aplicarea mecanismului de despachetare a unui tuplu (tuple unpacking) pentru extragerea valorilor (de exemplu, integrarea într-o aplicație de gestionare a elevilor, în care funcțiile returnează mai multe informații)* - *aplicarea metodelor și funcțiilor uzuale pentru tupluri în rezolvarea unor probleme practice, de exemplu, pentru gestionarea unui magazin, în care încasările săptămânale sunt memorate în tupluri, și pe baza acestora se determină numărul total de zile cu încasări, ziua cu cele mai multe/puține încasări, suma totală încasată pe săptămână, încasările sortate crescător/descrescător, verificarea existenței a cel puțin unei zile cu încasări, verificarea faptului că au fost încasări în fiecare zi înregistrată*
CS 3.5. Utilizează elementele de sintaxă și componentele din definiția și apelul subprogramelor recursive, în implementarea algoritmilor în limbaj de programare
- *aplicarea elementelor de sintaxă pentru implementarea unui subprogram recursiv, pe baza unei relații recurente date, care determină factorialul unui număr, de exemplu, pentru stabilirea numărului de moduri diferite în care pot fi așezate 4 persoane pe 4 scaune într-o sală* - *aplicarea recursivității pentru determinarea unui termen din șirul lui Fibonacci, prin urmărirea apelurilor unui subprogram recursiv în Python* - *aplicarea recursivității pentru calcularea sumei elementelor unei liste, pe baza unui subprogram recursiv dat în limbajul Python, care adună elementul curent al listei cu rezultatul obținut prin apelarea sa asupra restului listei* - *implementarea unui subprogram recursiv pornind de la o relație de recurență dată, de exemplu, pentru calculul combinărilor de câte k elemente dintre cele n date, pe baza formulei $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ cu respectarea condițiilor matematice*

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

Clasa a X-a
CS 4.1. Analizează caracteristicile, modul de prelucrare, avantajele și dezavantajele utilizării unor modele conceptuale simple - neliniare, liniare, asociative, mixte, pentru a alege structura adecvată în rezolvarea unor probleme concrete
- *analizarea avantajelor și limitărilor utilizării mulțimilor în comparație cu listele, prin raportare la situații practice, precum verificarea rapidă a înscrierii unui elev în catalogul clasei, cu evidențierea faptului că verificarea apartenenței unui element este mai eficientă din punctul de vedere al numărului de operații efectuate, utilizând mulțimi comparativ cu liste, dar acestea nu păstrează ordinea și nu acceptă elemente duplicat* - *compararea diferitelor modalități de prelucrare a unui șir de caractere care conține numele urmat de toate prenumele unei persoane, prin separarea acestor componente pentru a le reuni într-un nume format din toate prenumele urmate de nume* - *analizarea modelelor de organizare a datelor care să permită prelucrarea unui mesaj real, precum un e-mail sau un anunț online, pentru a separa informațiile esențiale (expeditorul, destinatarul, subiectul și mesajul principal)* - *analizarea avantajelor utilizării unui dicționar într-o aplicație financiară, în care codul fiscal al fiecărui client este utilizat drept cheie pentru accesarea directă a datelor asociate, astfel încât informațiile despre un client sunt obținute imediat, fără a fi nevoie de verificarea succesivă a tuturor celorlalți clienți din evidență* - *analizarea comportamentului datelor organizate în dicționare la actualizarea valorilor pentru chei existente versus chei noi, în contextul modificării prețului unui produs existent (cheie existentă) sau la adăugarea unui produs nou (cheie nouă) într-un magazin* - *analizarea modului în care este utilizat un dicționar într-un sistem de evidență al unei biblioteci, în care fiecare carte este identificată printr-o cheie (de exemplu, codul cărții), iar valoarea asociată indică plasarea acesteia, pentru a verifica dacă fiecărei chei îi corespunde o plasare, evitând astfel pierderea informației despre plasarea cărților în bibliotecă* - *compararea a două moduri de stocare a datelor despre produse – într-o listă de liste (unde poziția contează) și într-o listă de dicționare (unde accesul se face prin chei), pentru a alege varianta mai clară și mai flexibilă* - *analizarea adecvării utilizării tuplurilor, pentru a organiza datele (disciplinele) în cadrul unui program săptămânal fixat* - *examinarea modului în care folosirea tuplurilor într-o aplicație medicală contribuie la păstrarea neschimbată a datelor esențiale ale unui pacient (nume, CNP, data nașterii), prin utilizarea unei structuri cu valori fixe care limitează alterarea accidentală a informațiilor* - *analizarea modului în care perechile (produs, preț) sunt memorate sub formă de tupluri într-un sistem de vânzări, pentru a verifica dacă prețul asociat fiecărui produs este corect utilizat în prelucrări precum afișarea costului unui articol, calculul valorii totale a unei comenzi sau compararea prețurilor între produse*
CS 4.2. Analizează comportamentul, aplicabilitatea, avantajele și dezavantajele utilizării unor algoritmi specializați pe clase de probleme pentru prelucrarea listelor, a șirurilor de caractere
- *examinarea modului în care metoda căutării binare restrânge treptat intervalul de căutare într-o listă ordonată, prin raportare la situații reale precum găsirea unui zbor într-o listă de plecări ordonată după oră sau localizarea unui elev într-un registru ordonat după numărul matricol* - *compararea modului de funcționare și a numărului de pași necesari în căutarea secvențială față de căutarea binară, pentru a evidenția diferențele de eficiență în funcție de dimensiunea listei* - *analizarea situațiilor în care metoda căutării binare nu este utilizată adecvat, prin observarea greșelilor frecvente precum aplicarea metodei în cazul unor elemente neordonate, condiții greșite sau actualizarea incorectă a limitelor de căutare* - *compararea numărului de operații necesare pentru a combina listele de rezervări provenite de la două hoteluri ale aceluiași lanț, fiecare listă fiind ordonată după același criteriu, de exemplu, data sosirii clienților, pentru a obține o singură listă finală, cu toți clienții, ordonată cronologic după momentul sosirii, prin metoda interclasării, respectiv prin concatenarea celor două liste și ordonarea listei obținute* - *analizarea erorilor care pot apărea în aplicarea metodei interclasării, precum omiterea unor elemente din una dintre liste, includerea aceluiași element de mai multe ori sau pierderea ordinii atunci când elementele nu sunt comparate corect după criteriul stabilit* - *compararea metodelor de criptare studiate (cifrul lui Cezar, substituția simplă, cifrul Vigenère) din perspectiva modului de transformare a mesajului, a tipului de cheie utilizate și a nivelului de dificultate în descifrarea unui mesaj real, precum transmiterea unui mesaj confidențial între două persoane* - *analizarea mesajelor criptate obținute în situații concrete cu metodele studiate, precum trimiterea unui cod de acces sau a unui mesaj privat, pentru a identifica erori de criptare, cum ar fi litere transformate incorect, pierderea unor caractere sau aplicarea greșită a regulilor metodei alese*

Clasa a X-a
- *examinarea asemănărilor și deosebirilor dintre metodele de criptare utilizate în contexte reale, precum comunicarea mesajelor secrete sau protejarea informațiilor scrise, prin raportare la modul de transformare a literelor, dependența de cheie și ușurința cu care mesajul poate fi descifrat*
CS 4.3. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor
- *analizarea aplicării metodei Divide et impera printr-un exemplu concret, în care o listă de scoruri ale jucătorilor este împărțită repetat în jumătate, până când se ajunge la liste formate dintr-un singur scor, fiecare sublistă este prelucrată separat prin aceeași regulă (de exemplu, determinarea scorului maxim/minim din sublistă), iar rezultatele obținute sunt apoi comparate pentru a stabili scorul maxim din lista inițială, evidențiind explicit etapele de împărțire, rezolvare și combinare* - *analizarea procesului de împărțire succesivă a unei probleme prin metoda Divide et impera, printr-un exemplu precum calculul sumei elementelor unei liste* - *analizarea modului în care aplicarea succesivă a deciziilor locale din metoda Greedy conduce la obținerea unei soluții finale concrete, de exemplu, selectarea unui set de activități dintr-o mulțime de activități dată, care să nu se suprapună în timp, astfel încât numărul total de activități alese să fie cât mai mare, evidențiind legătura dintre fiecare alegere intermediară și rezultatul final obținut* - *analizarea modului în care metoda Greedy construiește o soluție pas cu pas într-un context real, precum alegerea unor produse (care pot fi preluate și parțial) într-un buget limitat, urmărind cum fiecare alegere locală (de exemplu, selectarea produsului cu cel mai bun raport preț–valoare la acel moment) conduce la obținerea unei liste finale de produse selectate, pentru care se poate calcula explicit valoarea totală obținută, reprezentând rezultatul urmărit al aplicării metodei* - *analizarea situațiilor reale în care metoda Greedy conduce la un rezultat necorespunzător, cum ar fi oferirea restului cu monede de valori diferite, evidențiind cazurile în care alegerea repetată a celei mai mari monede disponibile nu conduce la o soluție*
CS 4.4. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor elemente ale limbajului de programare, pentru a identifica soluția adecvată prelucrării datelor organizate în modele simple - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice
- *analizarea mesajelor de eroare apărute în Python la includerea într-o mulțime a unor elemente mutabile (de exemplu, liste sau dicționare), prin asocierea cu situația în care într-o colecție de obiecte unice se încearcă adăugarea unui obiect care își poate schimba forma (de exemplu, o trusă de geometrie ce poate conține doar reprezentări ale unor corpuri geometrice care sunt rigide, deci nu se pot modifica, comparativ cu cazul în care acestea ar fi realizate din plastilină, astfel încât să poată fi remodelate)* - *analizarea comportamentului unor programe care realizează operații de bază cu mulțimi (de exemplu, verificări rapide ale apartenenței, identificarea dublurilor într-o listă de înscrieri), utilizând clasa set, respectiv list, pentru alegerea unei variante eficiente din punctul de vedere al timpului de executare* - *analizarea avantajelor utilizării unei liste de șiruri de caractere, comparativ cu un șir compact, pentru afișarea în ordine alfabetică a unor cuvinte* - *analizarea contextelor de prelucrare a șirurilor de caractere, de exemplu, pentru verificarea și corectarea mesajelor trimise într-o conversație online sau într-un document școlar, unde pot apărea greșeli cauzate de folosirea incorectă a ghilimelelor/apostrofurilor la definirea unui șir, utilizarea unor metode (split(), replace(), strip()) pentru date care nu sunt de tip str, confuzia dintre șiruri de caractere și numere în timpul procesării unui text sau încercarea de a modifica un șir, deși acesta este imutabil* - *analizarea avantajelor utilizării dicționarelor în contextul comparării stocurilor a două magazine pentru a identifica produse disponibile în ambele magazine (chei comune), produse care există doar într-un magazin și nu în celălalt (diferențe)* - *analizarea modului în care pot fi combinate dicționarele, în contextul unificării stocurilor din mai multe magazine, pentru adăugarea sau actualizarea datelor produselor din magazinul principal (update()), unificarea stocurilor pentru generarea unui catalog de produse (operatorii "|", "**"), unificarea manuală și verificarea cheilor duplicate, filtrarea produselor cu preț mai mare decât o valoare dată în momentul unificării* - *analizarea avantajelor bordării pentru memorarea datelor în Python, organizate sub forma unui model mixt, de tipul listă de liste, pentru determinarea numărului de elemente nenule care au doar vecini nuli* - *analizarea necesarului de memorie utilizată de un tablou bidimensional în C++ de dimensiuni mari, pentru memorarea datelor organizate sub forma unui model mixt, de tipul listă de liste, pentru reprezentarea unui labirint în care zidurile sunt marcate cu 1, iar culoarele cu 0* - *analizarea avantajelor utilizării unui tuplu care conține obiecte mutabile (liste, dicționare), în contextul reținerii și prelucrării datelor despre produsele dintr-un magazin într-un tuplu de liste, de exemplu, magazin=([‘paine’, 20], [‘lapte’, 15], [‘mere’, 30]), unde pe prima poziție a fiecărei liste este denumirea produsului, iar pe a doua poziție este cantitatea aferentă*

Clasa a X-a
CS 4.5. Analizează aplicabilitatea, avantajele și dezavantajele utilizării subprogramelor recursive, în implementarea algoritmilor în limbaj de programare
- *analizarea consumului de timp și memorie al unui subprogram recursiv în Python care rezolvă o problemă matematică, precum determinarea sumei primelor n numere naturale, prin observarea numărului de apeluri recursive efectuate și a spațiului ocupat în stiva de executare pe măsură ce parametrul utilizat se modifică* - *compararea abordării recursive și a celei iterative în Python pentru rezolvarea unei probleme matematice precum calculul sumei primelor n numere naturale, analizând diferențele dintre cele două moduri de rezolvare în ceea ce privește numărul de pași efectuați, modul de repetare a operațiilor și utilizarea memoriei în funcție de valoarea lui n* - *compararea a doi algoritmi pentru determinarea unui termen al șirului lui Fibonacci, unul iterativ, iar celălalt recursiv, pe baza desfășurării pas cu pas a executării acestuia, urmărind ordinea apelurilor și revenirea din acestea, prin analogie cu verificarea succesivă a unor cutii puse una în alta, unde fiecare verificare nouă este memorată temporar până când se ajunge la ultima cutie și se revine treptat la prima, observând reluarea unor calcule*

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

Clasa a X-a
CS 5.1. Argumentează alegerea unor modele conceptuale simple - neliniare, liniare, asociative, mixte și a modurilor de prelucrare a acestora pentru rezolvarea unor probleme informatice concrete
- *argumentarea adecvării utilizării mulțimilor care gestionează date neordonate și fără elemente duplicate (de exemplu, administrarea codurilor unice ale produselor dintr-un magazin, evidența participanților la un eveniment) cu justificarea alegerii acestora pentru avantajele de unicitate* - *evaluarea avantajelor utilizării mulțimilor în organizarea unui volum mare de informații despre clienți, în comparație cu alte forme de organizare precum listele, justificând alegerea mulțimilor prin capacitatea lor de a elimina automat aparițiile repetate și de a permite identificarea rapidă a elementelor unice, ceea ce reduce erorile și timpul de prelucrare a datelor* - *evaluarea eficienței unei proceduri de extragere a informațiilor din fișiere text/documente, precum preluarea automată a numelor, codurilor sau valorilor dintr-un fișier, justificând dacă metoda utilizată reduce timpul de procesare și minimizează erorile față de o prelucrare manuală* - *argumentarea adecvării utilizării șirurilor de caractere pentru reprezentarea și prelucrarea informațiilor textuale (de exemplu, coduri de identificare, adrese de e-mail sau nume de utilizatori), cu justificarea alegerii acestui tip de date prin avantajele legate de lizibilitate, posibilitatea validării caracter cu caracter și identificarea clară a informațiilor* - *evaluarea corectitudinii utilizării unui dicționar digital într-o aplicație reală (de exemplu, un catalog electronic de produse), prin verificarea faptului că fiecare cheie este unică și că valoarea asociată conține informații complete și corecte, justificând că această structură previne confuziile și erorile de afișare în timpul utilizării* - *justificarea alegerii dicționarului pentru rezolvarea unor probleme din context cotidian (de exemplu, gestionarea unui catalog de elevi, a unei biblioteci, a unui concurs sportiv, a unei agenții de turism, organizarea informațiilor economice cum ar fi asocierea codurilor de factură cu datele corespunzătoare ale clienților), prin accesul rapid la date și prin claritatea relației dintre identificator și informația asociată* - *evaluarea avantajelor utilizării unei structuri de tip dicționar care asociază termeni cu definițiile lor (de exemplu, un glosar digital), prin aprecierea corectitudinii relației termen–definiție și justificarea modului în care această organizare sprijină regăsirea rapidă și fără ambiguități a informațiilor* - *evaluarea corectitudinii datelor într-un model conceptual mixt (de exemplu, listă de liste, listă de dicționare) pentru un sistem de rezervări care verifică dacă toate listele de locuri disponibile corespund cu realitatea, detectând eventualele inconsistențe (de exemplu, locuri duble sau lipsă)* - *argumentarea alegerii unui model mixt pentru un sistem de catalog electronic (de exemplu, listă de liste, listă de dicționare) pentru a organiza datele despre clase, elevi și note, asigurând o structură clară și flexibilă* - *evaluarea adecvării utilizării tuplurilor pentru memorarea unor date care nu se modifică, precum coordonatele geografice ale unui punct sau datele de identificare ale unui document, justificând alegerea prin stabilitatea valorilor și reducerea riscului de modificări accidentale* - *justificarea alegerii unui tuplu în locul unei liste pentru a organiza date constante (de exemplu, coordonatele unui punct fix pe hartă, programul fix al săptămânii, coduri de culori RGB etc.)*
CS 5.2. Evaluează corectitudinea și eficiența soluțiilor cu algoritmi specializați pe clase de probleme pentru prelucrarea listelor, a șirurilor de caractere
- *evaluarea corectitudinii unei soluții în care sunt căutați, în mod repetat, elevii dintr-o listă ordonată alfabetic, justificând eliminarea căutărilor inutile în una dintre jumătățile listei* - *evaluarea eficienței integrării metodei căutării binare într-o aplicație funcțională, prin justificarea modului în care aceasta permite localizarea rapidă a unei informații într-o listă ordonată*

Clasa a X-a
- *evaluarea eficienței algoritmului de interclasare a listelor ordonate în contexte reale, precum afișarea tuturor produselor unui magazin online, în ordine crescătoare după preț, pe baza a două astfel de liste, una cu produse alimentare, și alta cu produse nealimentare, comparativ cu concatenarea celor două liste și ordonarea celei obținute* - *evaluarea adecvării metodelor de criptare și decriptare utilizate într-un context real, precum protejarea mesajelor transmise într-o aplicație de comunicare, prin justificarea alegerii metodei (cifrul lui Cezar, substituție simplă, cifrul Vigenère) în funcție de nivelul de securitate necesar, tipul cheii folosite și ușurința de aplicare* - *evaluarea eficienței utilizării metodelor de criptare într-un context real de transmitere a documentelor, de exemplu, trimiterea prin e-mail a unui fișier cu date personale, prin analizarea modului în care textul este criptat cu cifrul Vigenère pentru confidențialitate, în vederea justificării faptului că informația este protejată împotriva accesului neautorizat*
CS 5.3. Evaluează corectitudinea și eficiența algoritmilor care utilizează strategii de tip Divide et impera, Greedy pentru rezolvarea problemelor
- *evaluarea eficienței metodei Divide et impera în rezolvarea unor probleme cotidiene (de exemplu, sortarea unui șir de melodii după durată sau căutarea rapidă a unui artist într-o listă ordonată alfabetic), prin justificarea alegerii metodei pe baza timpului de executare, a clarității pașilor și a comparării cu alte metode de rezolvare* - *evaluarea oportunității aplicării metodei Divide et impera în contexte matematice (de exemplu, căutarea unui element într-un interval numeric prin înjumătățiri succesive), prin justificarea deciziei pe baza modului de împărțire a problemei în subprobleme independente, a reducerii numărului de pași și a eficienței soluției finale* - *evaluarea corectitudinii unui algoritm bazat pe metoda Divide et impera utilizat într-o problemă matematică (de exemplu, calculul rapid al unei puteri sau al maximului dintr-un șir de valori), prin justificarea validității soluției obținute pentru toate subproblemele și a modului în care acestea sunt combinate* - *evaluarea modului în care optimul local în metoda Greedy conduce la o soluție eficientă în situații practice (de exemplu, selectarea produselor cu cel mai mare raport valoare/preț într-un buget limitat), prin justificarea faptului că ordonarea datelor și selecția pas cu pas a optimului local conduc la o soluție viabilă* - *evaluarea obținerii soluției optime pentru probleme rezolvate cu metoda Greedy (de exemplu, organizarea unui program de întâlniri fără suprapuneri), prin justificarea corectitudinii condițiilor alese* - *argumentarea deciziei de aplicare a metodei Greedy în contexte matematice (de exemplu, realizarea unei sume maxime cu n termeni de forma a_i*b_j unde $a_i \in A$, $b_j \in B$, iar A și B sunt mulțimi de numere întregi, $card(A)=n$ și $card(B) \geq n$), cu justificarea eficienței obținute ca urmare a luării deciziilor locale și a respectării condițiilor de aplicabilitate ale metodei*
CS 5.4. Evaluează corectitudinea și eficiența aplicațiilor care utilizează elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele simple - neliniare, liniare, asociative, mixte, în vederea rezolvării eficiente a problemelor informatice
- *evaluarea corectitudinii și eficienței diferitelor soluții de eliminare a elementelor duplicate dintr-o listă, prin compararea implementărilor iterative cu cele bazate pe conversia listei într-o mulțime (set), pentru a evidenția avantajele utilizării mulțimilor în simplificarea și accelerarea procesului de filtrare* - *evaluarea eficienței a două soluții pentru aceeași problemă, una care integrează operații specifice clasei set pentru îmbunătățirea clarității și performanței și alta bazată pe liste (de exemplu, gestionarea datelor de înscriere la un eveniment, filtrarea elementelor unice dintr-un fișier)* - *evaluarea eficienței și clarității redactării a două soluții pentru aceeași problemă, care verifică dacă un cuvânt este anagramă a unui alt cuvânt* - *argumentarea alegerii unei metode din clasa str pentru o sarcină practică, de exemplu, căutarea unor vocale într-un text dat* - *evaluarea eficienței unei soluții bazate pe dicționar față de una bazată pe listă de liste (listă de perechi), în contextul căutării rapide a prețului unui produs într-un catalog* - *evaluarea unui cod care folosește inadecvat metodele clasei dict (de exemplu, încercarea de a accesa o cheie inexistentă fără metoda get(), utilizarea metodei pop() cu argument greșit, încercarea de a modifica o cheie etc.)* - *evaluarea corectitudinii expresiilor Python utilizate pentru accesarea vecinilor unui element în cazul memorării datelor organizate sub forma unui model mixt, de tipul listă de liste, în contextul unei fotografii alb-negru în care punctele corespunzătoare obiectelor sunt marcate cu 1, iar punctele de fundal cu 0* - *evaluarea corectitudinii declarării unei variabile de tip tablou bidimensional în C++ din punctul de vedere al necesarului de memorie utilizată, pentru memorarea datelor organizate sub forma unui model mixt, de tipul listă de liste, pentru modelarea unei hărți în care zonele de uscat sunt marcate cu 1, iar cele de apă cu 0* - *evaluarea clarității unui cod care folosește tupluri pentru transmiterea mai multor valori dintr-o funcție (de exemplu, returnarea simultană a datelor despre un produs dintr-un magazin, cum ar fi denumire, preț, cantitate)* - *evaluarea corectitudinii unui cod în care sunt utilizate tupluri, având în vedere enunțul problemei corespunzătoare și corectarea eventualelor greșeli*

Clasa a X-a
CS 5.5. Evaluează corectitudinea și eficiența programelor care utilizează subprograme recursive, pentru a rezolva probleme informatice
- *evaluarea eficienței unui subprogram recursiv în Python în raport cu o implementare iterativă echivalentă pentru o problemă matematică (de exemplu, calculul factorialului unui număr), prin justificarea diferențelor de claritate a soluției, număr de pași de executare și consum de memorie* - *evaluarea limitărilor în utilizarea unui subprogram recursiv în Python pentru rezolvarea unei probleme matematice (de exemplu, determinarea celui de al n-lea termen al şirului lui Fibonacci), evidenţiind apeluri repetate pentru realizarea unor calcule identice* - *evaluarea deciziei de folosire a unui subprogram recursiv în Python pentru organizarea unui buget complex (de exemplu, calcularea soldului lunar pentru un buget structurat ierarhic, folosind liste de liste sau dicționare de dicționare), prin justificarea modului în care structura recursivă simplifică tratarea nivelurilor multiple de date față de o soluție iterativă*

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

Clasa a X-a
CS 6.1. Proiectează algoritmi personalizați care utilizează modele conceptuale simple - neliniare, liniare, asociative, mixte, pentru organizarea și prelucrarea eficientă a datelor, utilizând algoritmi de bază, în vederea rezolvării unor probleme
- *proiectarea unui algoritm de păstrare a evidențelor pentru o organizație educațională (de exemplu, o platformă de cursuri), pentru a identifica participanții înscriși, a evita înscrierile redundante, pe baza datelor utilizatorilor care sunt organizate în mulțimi* - *proiectarea unui algoritm pentru analizarea datelor unui magazin online, în care sunt definite explicit mulțimea tuturor clienților înregistrați și, pentru fiecare produs, mulțimea clienților care l-au achiziționat, iar relațiile dintre mulțimi sunt utilizate pentru recomandarea de perechi de produse care sunt achiziționate de obicei împreună* - *proiectarea unui algoritm care validează datele de tip text introduse într-un formular (nume, CNP, adresă de e-mail), prin definirea și aplicarea unor reguli clare, precum lungimea minimă și maximă a șirurilor de caractere, tipul caracterelor permise, structura obligatorie a datelor și verificarea formatului corect, cu afișarea de mesaje de eroare personalizate atunci când regulile nu sunt respectate* - *crearea unui scenariu de analiză automată a textelor educaționale, care parcurge șirurile de caractere dintr-un eseu pentru a calcula frecvența cuvintelor și a genera feedback privind diversitatea vocabularului utilizat* - *proiectarea unui algoritm de actualizare automată a unui dicționar de cursuri valutare, în care cheia este codul monedei (de exemplu, EUR, USD), iar valoarea este cursul de schimb asociat în lei, algoritmul preluând periodic date noi și înlocuind valorile existente cu cele actualizate* - *proiectarea unui sistem de înlocuire automată a termenilor dintr-un text, folosind un dicționar în care cheia este cuvântul sau expresia căutată, iar valoarea este cuvântul sau expresia care o înlocuiește, algoritmul parcurgând textul și substituind fiecare apariție a cheilor cu valorile corespunzătoare* - *proiectarea unui algoritm care utilizează un model mixt (listă de liste sau dicționar de liste) pentru un sistem de evidență a opțiunilor angajaților unei companii, în care există o ofertă de 10 beneficii și fiecare angajat poate opta pentru maximum cinci dintre acestea, cu posibilitatea afișării angajaților care au optat pentru un anumit beneficiu, a beneficiilor pentru care a optat fiecare angajat* - *proiectarea unui algoritm de filtrare a comenzilor dintr-o listă în care pentru fiecare înregistrare se păstrează suma plătită și numărul bonului, pentru a afișa doar comenzile care depășesc o sumă prestabilită* - *proiectarea unui algoritm bazat pe tupluri pentru afișarea clasamentelor sportive, în care fiecare tuplu conține numele jucătorului, punctajul obținut și poziția finală, astfel încât datele să corespundă exact rezultatelor stabilite la finalul competiției și să nu poată fi modificate ulterior, păstrând coerența datelor pe toată durata competiției* - *proiectarea unui algoritm pentru înregistrarea itinerariului parcurs de un grup de turiști, utilizând tupluri, fiecare conținând coordonatele GPS și denumirea obiectivului vizitat*
CS 6.2. Proiectează soluții cu aplicarea personalizată a algoritmilor specializați pe clase de probleme pentru prelucrarea listelor, a șirurilor de caractere
- *crearea unei aplicații funcționale care integrează căutarea binară, utilizată pentru gestionarea unor date reale (de exemplu, identificarea rapidă a datelor de contact ale unui client într-o listă ordonată a acestora sau a unui produs într-o listă ordonată de inventar), evidențiind modul în care ordonarea datelor permite căutarea eficientă* - *proiectarea unor soluții informatice care utilizează interclasarea pentru situații din viața reală, precum gestionarea programărilor medicale realizate independent online, respectiv telefonic, prin combinarea și accesarea eficientă a datelor* - *proiectarea unor algoritmi pentru prelucrarea listelor ordonate, aplicați în contexte reale precum combinarea listelor de produse din mai multe depozite, fiecare ordonată alfabetic, prin utilizarea metodei de interclasare pentru a obține o listă finală ordonată alfabetic*

Clasa a X-a
- *proiectarea unor soluții informatice pentru protejarea mesajelor transmise într-un context real (de exemplu, comunicarea dintre membrii unei echipe sau stocarea unor note personale), prin proiectarea unor aplicații care implementează criptarea și decriptarea șirurilor de caractere folosind cifrul lui Cezar sau cifrul Vigenère, explicând modul în care cheia și regulile de transformare asigură confidențialitatea informației* - *proiectarea unor soluții pentru verificarea integrității datelor textuale (de exemplu, validarea unui fișier trimis online sau detectarea modificărilor unui mesaj), prin aplicarea algoritmului lui Fletcher, astfel încât soluția să fie mai eficientă și mai sigură în raport cu problema reală rezolvată*
CS 6.3. Proiectează algoritmi cu aplicarea personalizată a strategiilor de tip Divide et impera, Greedy pentru rezolvarea problemelor
- *proiectarea unor algoritmi care utilizează metoda Divide et impera, pentru contexte reale (de exemplu, căutarea unui document într-o arhivă ordonată), prin definirea adecvată a datelor și a subproblemelor, astfel încât rezultatul să fie obținut împărțind problema inițială în subprobleme mai mici de același tip* - *proiectarea unor algoritmi pentru crearea unor fractali simpli (de exemplu, covorul lui Sierpinski), utilizând metoda Divide et impera* - *proiectarea unor soluții informatice care aplică personalizat metoda Divide et impera pentru problema turnurilor din Hanoi prin reprezentarea configurației tijelor folosind structuri de date adecvate și prin afișarea etapizată a mutărilor efectuate* - *proiectarea unui algoritm cu metoda Greedy pentru obținerea unei soluții finale concrete, de exemplu, selectarea unui set de activități dintr-o mulțime de activități dată, care nu se suprapun în timp, astfel încât numărul total de activități alese să fie cât mai mare, prin stabilirea unui criteriu de alegere locală și aplicarea succesivă a acestuia pentru obținerea unei soluții finale* - *proiectarea unui algoritm cu metoda Greedy pentru un context real, precum alegerea unor produse într-un buget limitat, urmărind cum fiecare alegere locală (de exemplu, selectarea produsului cu cel mai bun raport preț–valoare la acel moment, cu posibilitatea preluării parțiale a acestuia) conduce la obținerea unei liste finale de produse selectate, pentru care se poate calcula explicit valoarea totală obținută, reprezentând rezultatul urmărit al aplicării metodei*
CS 6.4. Implementează aplicații care integrează în mod personalizat elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele simple - neliniare, liniare, asociative, mixte, în vederea realizării unor soluții funcționale și performante
- *implementarea unor aplicații personalizate care utilizează clase Python specifice pentru gestionarea mulțimilor la matematică (de exemplu, un software educațional cu titlul „Mulțimi”) sau generarea unor numere distincte pentru jocuri de tip Bingo* - *implementarea unor aplicații personalizate pentru prelucrarea cifrelor unui număr utilizând adecvat clasa set din Python (de exemplu, determinarea cifrelor distincte ale unui număr, formarea unui număr maxim cu cifrele distincte ale unui număr dat, determinarea cifrelor comune ale unor numere date, determinarea cifrelor care nu apar în scrierea unui număr etc.)* - *implementarea unor programe personalizate folosind metode adecvate din clasa Python str pentru prelucrarea șirurilor de caractere, pentru codificarea unui text cu o regulă de criptare simplă (de exemplu, prin înlocuirea fiecărei litere cu următoarea literă din alfabet, cu excepția literei ‘z’, care se înlocuiește cu litera ‘a’)* - *implementarea unor programe personalizate folosind funcționalitățile clasei str din Python, pentru verificarea formatului unei adrese de e-mail (dacă adresa conține exact un caracter ‘@’ și cel puțin un caracter ‘.’ după acesta, dar nu conține spații)* - *implementarea unor programe personalizate folosind funcționalitățile clasei dict, pentru gestiunea stocurilor dintr-un magazin* - *implementarea unor programe personalizate folosind funcționalitățile clasei dict, pentru traducerea automată a unor cuvinte în mai multe limbi* - *implementarea unor programe Python în care se memorează date organizate sub forma unui model mixt, de tipul listă de liste, corespunzătoare unei table de șah, cu toate piesele regine, în care căsuțele cu piese albe sunt marcate cu 1, căsuțele cu piese negre sunt marcate cu -1, iar căsuțele libere sunt marcate cu 0, pentru determinarea numărului de piese atacate de o regină de altă culoare, aflată la o poziție dată* - *implementarea unor programe C++ în care se memorează, într-un tablou bidimensional, date organizate sub forma unui model mixt, de tipul listă de liste, corespunzătoare unui covor de formă dreptunghiulară, în care fiecare element are o valoare ce depinde de culoarea zonei corespunzătoare, pentru a verifica simetria stânga-dreapta a modelului de pe covor* - *integrarea tuplurilor în structuri mixte, cum ar fi liste de tupluri (de exemplu, pentru gestionarea stocului de produse într-un magazin, reprezentat sub formă de listă, în care fiecare produs este reținut printr-un tuplu care conține date fixe, cum ar fi codul produsului, denumirea, prețul), dicționare în care valorile sunt tupluri (de exemplu, catalog în care cheile sunt reprezentate de numele elevilor, iar valorile sunt notele fiecărui elev, reținute în tupluri)*

Clasa a X-a
- *implementarea unor aplicații care folosesc tupluri, de exemplu, pentru gestionarea punctajelor la un concurs, unde fiecare tuplu conține (nume, scor), pentru gestionarea coordonatelor geografice pentru un explorator sau pentru gestionarea codurilor de culori RGB pentru imagini*
CS 6.5. Implementează aplicații în limbaj de programare care integrează subprograme recursive personalizate, pentru a modulariza și organiza eficient codul în cadrul soluțiilor informatice
- *crearea unei aplicații pentru prelucrarea cifrelor unui număr care afișează mai multe opțiuni, iar pentru fiecare opțiune se apelează câte un subprogram recursiv (de exemplu, pentru afișarea cifrelor în ordine de la stânga la dreapta, de la dreapta la stânga, determinarea cifrei maxime, a numărului de cifre, a sumei cifrelor pare, verificarea existenței unei cifre pare, verificarea proprietății de platou format cu cifre egale), fiecare subprogram fiind elaborat de câte un membru al unei echipe de elevi* - *crearea unei aplicații cu subprograme recursive pentru prelucrarea unei liste ordonate de numere (de exemplu, mediile unor elevi), prin care să se afișeze numerele atât în ordine crescătoare, cât și în ordine descrescătoare* - *crearea unui subprogram recursiv pentru a verifica dacă un cuvânt este o anagramă a unui alt cuvânt, în contextul unei aplicații de jocuri enigmistice* - *crearea unei aplicații cu subprograme recursive care să afișeze pe ecran diverse forme geometrice simple cu ajutorul spațiilor și al unor simboluri (de exemplu, un triunghi format din simboluri *), în contextul unui concurs de realizare a designului unei sigle*

CONȚINUTURI ALE ÎNVĂȚĂRII

Clasa a X-a

Domenii de conținut	Conținuturi
1. Organizarea conceptuală a datelor	1.1. Modelul conceptual neliniar - mulțime - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.2. Modelul conceptual liniar – șir de caractere - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.3. Modelul conceptual asociativ – dicționar - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.4. Modelul conceptual liniar - tuplu - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază în scopul prelucrării acestora. 1.5. Modelul conceptual mixt (de exemplu, liste de liste, liste de șiruri de caractere) - caracteristici și repere pentru acces la date, parcurgerea lor și pentru aplicarea algoritmilor de bază sau specifici în scopul prelucrării acestora (de exemplu, actualizarea datelor, modificarea numărului de componente ale colecției, construirea colecției).
2. Strategii de rezolvare a problemelor	2.1. Metode de prelucrare a listelor sortate - caracteristici ale metodei căutării binare și repere pentru aplicarea acesteia; - caracteristici ale metodei interclasării și repere pentru aplicarea acesteia. 2.2. Metode simple de criptare a șirurilor de caractere și verificare a integrității datelor - caracteristici ale criptării și decriptării folosind cifrul lui Cezar și repere pentru aplicarea metodei; - caracteristici ale criptării și decriptării prin metoda substituției simple (substituție monoalfabetică); - caracteristici ale criptării și decriptării utilizând cifrul Vigenère și repere pentru aplicarea metodei; - caracteristici ale sumei de control a unui șir de caractere sau a unui fișier text și determinarea sa utilizând algoritmul lui Fletcher. 2.3. Metoda Divide et impera - caracteristici ale metodei Divide et impera; - repere pentru aplicarea metodei; - caracteristici ale metodei de sortare prin interclasare și repere pentru aplicarea acesteia; - caracteristici ale metodei de sortare rapidă și repere pentru aplicarea acesteia. 2.4. Metoda Greedy - caracteristici ale metodei Greedy; - repere pentru aplicarea metodei.
3. Memorarea datelor și organizarea codului în limbaj de programare	3.1. Clasa set din Python – clasă predefinită pentru mulțimi - caracteristici; - operații și operatori specifici: reuniune, intersecție, diferență, egalitate, incluziune, apartenență; - metode uzuale pentru adăugare, ștergere, inițializare; - repere pentru identificarea și utilizarea altor metode adecvate pentru prelucrarea mulțimilor, în funcție de nevoile proprii. 3.2. Clasa str din Python – clasă predefinită pentru șiruri de caractere - caracteristici; - operații și operatori specifici: apartenență, concatenare, multiplicare, acces la un caracter sau subșir/secvență de caractere, comparare lexicografică; - metode uzuale pentru căutare a unei secvențe de caractere, generare a unui șir prin înlocuire a unei secvențe de caractere, separare a unor secvențe de caractere dintr-un șir; - repere pentru identificarea și utilizarea altor metode și funcții adecvate pentru prelucrarea șirurilor de caractere, în funcție de nevoile proprii. 3.3. Clasa dict din Python – clasă predefinită pentru dicționare - caracteristici; - operații și operatori specifici: acces la date;

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

SUGESTII METODOLOGICE

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

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

- centrare pe elev: strategiile să favorizeze implicarea activă a elevilor, de exemplu, învățarea prin descoperire, colaborarea și reflecția personală;
- diversitate metodologică: se recomandă utilizarea de metode variate;
- flexibilitate: profesorii adaptează activitățile de învățare la nivelul clasei și la resursele disponibile;
- corelare cu profilul de formare al absolventului: metodele didactice trebuie alese astfel încât să contribuie la formarea competențelor-cheie și a atributelor prioritare ale absolventului;
- integrare interdisciplinară: învățarea devine mai relevantă atunci când disciplinele se sprijină reciproc și creează punți între conținuturi;
- îmbinarea evaluării formative cu cea sumativă, cu recomandarea unor strategii de evaluare centrate pe o reflecție profundă asupra întrebărilor esențiale precum: De ce evaluez? Ce evaluez? Cum evaluez? Cât de bine 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 societale, să recunoască domeniile în care propria competență digitală trebuie îmbunătățită sau actualizată și să abordeze propriile nevoi în materie de competențe digitale în cadrul unui proces mai amplu de învățare pe tot parcursul vieții, de consolidare a capacităților și a autonomiei, oferind deschidere spre a îi sprijini și pe alții în dezvoltarea competențelor lor digitale.
Activitățile pe calculator sunt coordonate de profesor, care definește clar sarcinile, timpul alocat și criteriile de evaluare, adaptând nivelul de dificultate în funcție de particularitățile colectivului de elevi. Activitățile de învățare trebuie să fie alese adecvat, pentru

a contribui la formarea competențelor specifice din programă, astfel încât pentru nivelurile cognitive de recunoaștere și înțelegere se recomandă activități demonstrative și exerciții de identificare, pentru nivelul de aplicare se recomandă activități practice și aplicații asistate digital, pentru nivelurile de analiză și evaluare se recomandă studii de caz și proiecte interdisciplinare, iar pentru nivelul de creare se recomandă activități de tip învățare prin acțiune, realizarea de produse digitale și proiecte în echipă.

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

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

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

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

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

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

Un exemplu, orientativ, de ordine de abordare a conținuturilor pentru clasa a X-a, specializarea matematică-informatică militară

1. *Strategii de rezolvare a problemelor. Metode de prelucrare a listelor ordonate*
2. *Organizarea conceptuală a datelor. Modelul conceptual neliniar - mulțime*
3. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa set din Python*
4. *Organizarea conceptuală a datelor. Modelul conceptual liniar – șir de caractere*
5. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa str din Python*
6. *Strategii de rezolvare a problemelor. Metode simple de criptare a șirurilor de caractere și verificare a integrității datelor*
7. *Organizarea conceptuală a datelor. Modelul conceptual asociativ – dicționar*
8. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa dict din Python*
9. *Organizarea conceptuală a datelor. Modelul conceptual liniar – tupluri*
10. *Memorarea datelor și organizarea codului în limbaj de programare. Clasa tuple din Python*
11. *Organizarea conceptuală a datelor. Modelul conceptual mixt*
12. *Memorarea datelor și organizarea codului în limbaj de programare. Subprograme recursive*
13. *Strategii de rezolvare a problemelor. Metoda Divide et impera*
14. *Strategii de rezolvare a problemelor. Metoda Greedy*

Pentru prezentarea algoritmului de căutare binară profesorul poate pune accentul pe înțelegerea principiului de împărțire repetată a domeniului de căutare, implementarea algoritmului în Python și compararea căutării binare cu cea secvențială, utilizând metode didactice adecvate, ca problematizarea și descoperirea dirijată. Printre activitățile practice se poate prezenta o simulare vizuală pe tablă sau în aplicații online, astfel încât elevii să identifice mai ușor aspectele specifice metodei.
Se recomandă proiectarea algoritmului și reprezentarea acestuia ca schiță/în pseudocod, descrierea detaliată a etapelor rezolvării unei probleme din punctul de vedere algoritmic și apoi implementarea în limbaj de programare. Pentru a identifica elementele de eficiență caracteristice acestei metode se recomandă compararea unor algoritmi de căutare secvențială cu algoritmul de căutare binară, în scopul evidențierii eficienței căutării binare.
Pentru algoritmul de interclasare este necesar ca elevii să dețină deja competențe de bază privind parcurgerea și manipularea listelor simple și utilizarea indicilor. Interclasarea poate fi prezentată ca o metodă de combinare a două liste ordonate într-o listă unică, tot ordonată, nu ca o metodă de ordonare. Profesorul pune accentul pe logica algoritmică și pe relațiile de ordine dintre elementele listelor, evitând memorarea mecanică a pașilor: clarificarea condiției de aplicare (ambele liste să fie ordonate), stabilirea modului de parcurgere sincronă (utilizarea a doi indici care cresc independent), asigurarea completării finale (după ce una dintre liste s-a epuizat, elementele rămase din cealaltă se adaugă automat în lista finală), stabilitatea metodei (interclasarea păstrează ordinea relativă a elementelor egale - concept important pentru sortările stabile), ordinul de complexitate al algoritmului (liniar în raport cu numărul total de elemente O(n + m)).
Predarea conceptului de mulțime reprezintă o punte naturală între gândirea matematică și cea informatică. Se recomandă utilizarea strategiei descoperirii dirijate, pornind de la situații cotidiene, de exemplu, „mulțimea elevilor din clasă”, „mulțimea culorilor preferate”, pentru a stimula înțelegerea noțiunii de element și apartenență. Se pot folosi diagrame Venn pentru reprezentarea relațiilor de reuniune, intersecție și diferență, susținând astfel dezvoltarea gândirii vizuale și logice. Se recomandă integrarea metodei comparației pentru a reliefa caracterul neliniar al acestei structuri față de listele ordonate. Profesorul concretizează teoria mulțimilor prin clasa predefinită set. Se recomandă demonstrația dirijată și învățarea activă: elevii experimentează singuri comportamentul metodelor (add, remove, union, intersection, difference). Se valorifică interdisciplinaritatea cu matematica, punând accent pe semnificația logică a operațiilor.
Predarea șirurilor de caractere trebuie să urmărească trecerea de la conceptul abstract de „succesiune” la manipularea concretă a caracterelor. Inițial, conceptul de șir se poate introduce fără clasa str, la nivel de abstractizare logică: o succesiune de caractere ce pot fi accesate secvențial pornind de la observații intuitive („un cuvânt e un șir de litere”). Profesorul poate utiliza demonstrația explicativă, combinată cu descoperirea asistată – elevii analizează fragmente de cod Python ce ilustrează accesul la un caracter, concatenarea, căutarea unei secvențe de caractere. Metodologic, se recomandă învățarea colaborativă și feedbackul formativ, folosind instrumente digitale precum *Python Tutor* pentru vizualizarea pas cu pas a execuției codului. După înțelegerea conceptului, se trece la implementarea în Python cu str, pentru a exersa operațiile specifice (acces, concatenare, apartenență, căutare). Această etapă consolidează gândirea secvențială și logica parcurgerii.
Algoritmii de criptare au un rol formativ interdisciplinar, îmbinând concepte de informatică, logică, matematică și securitate a datelor. Tema oferă o introducere intuitivă în criptografie și în ideea de protejare a informației, consolidând totodată lucrul cu șiruri de caractere și prelucrări pe baza codurilor ASCII/Unicode.
Se urmărește formarea gândirii algoritmice aplicate și înțelegerea principiului de transformare a datelor (criptare ↔ decriptare), nu memorarea formulelor. Abordarea metodologică trebuie să fie intuitivă, progresivă și comparativă, pornind de la concepte simple (cifrul lui Cezar) către metode mai complexe (cifrul Vigenère), accentuând caracteristicile conceptuale și logice ale algoritmilor, principiile comune ale criptării: existența unui text clar și a unui text cifrat, utilizarea unei chei sau a unei reguli de transformare și posibilitatea de reversibilitate (criptare ↔ decriptare). Se recomandă utilizarea exemplelor concrete de mesaje și transformări simple pentru a formula regulile generale ale metodelor. Fiecare metodă se exprimă prin pași logici de prelucrare, accentuând legătura dintre idee și implementarea algoritmică și se valorifică legăturile cu alte domenii (matematică – operații modulo, statistică – frecvențe de apariție, TIC – fișiere text). Fiecare metodă poate fi analizată comparativ, în raport cu tipul transformării (substituție,

deplasare, control), complexitatea algoritmului și gradul de securitate indus, evidențiind asemănările și deosebirile dintre acestea (cifrul lui Cezar, cifrul Vigenère, substituție).
Profesorul poate introduce modelul asociativ, reprezentat de dicționar, prin metoda modelării, explicând principiul perechilor cheie–valoare și apelând la exemple familiare (agenda de contacte, glosar de termeni). Elevii sunt ghidați prin conversație euristică și comparare între listă și dicționar, pentru a înțelege diferențele privind accesul la date. Se recomandă integrarea live coding-ului și a vizualizării în timp real a modificării structurilor, consolidând astfel legătura între teorie și practică.
Abordarea modelului conceptual liniar - tuplu poate porni de la exemple care să ilustreze ideea de colecție ordonată și imutabilă, cum ar fi coordonatele unui punct fix pe o hartă sau programul constant al unei zile. Pentru o mai bună înțelegere, se pot realiza paralele între tupluri și liste, prin analogii concrete: tuplul ca o informație permanentă (de exemplu, locația unei clădiri), iar lista ca o informație care se poate modifica (de exemplu, poziția unui robot). Compararea comportamentului tuplurilor și al listelor, din perspectiva mutabilității, performanței și siguranței datelor, evidențiind avantajele folosirii tuplurilor pentru informații fixe, poate contribui la o mai bună înțelegere a conceptelor. Se recomandă utilizarea exemplelor vizuale și contextuale (coordonate, culori RGB, programul zilnic al unei clase etc.) pentru a evidenția modul de creare, accesare și parcurgere a elementelor unui tuplu, exerciții practice care implică prelucrarea datelor reale (de exemplu, reținerea și afișarea numărului de ore de activități zilnice, combinarea tuplurilor pentru generarea unui orar extins, extragerea valorilor multiple returnate de o funcție prin despachetare). Profesorul poate valorifica situații de învățare interdisciplinare (matematică, geografie, economie) pentru a consolida înțelegerea rolului tuplurilor în stocarea datelor constante.
Predarea modelelor mixte de organizare a datelor (liste de liste, liste de șiruri de caractere, liste de dicționare etc.) poate fi abordată gradual, pornind de la exemple simple și concrete (liste de liste, liste de dicționare), în care apare nevoia de ierarhizare și acces multiplu la date, către generalizări (dicționare de dicționare). Se recomandă folosirea situațiilor reale (tabele, cataloage, colecții de obiecte) pentru a evidenția avantajele fiecărei structuri. Pentru modelul conceptual mixt de tip listă de liste se recomandă abordarea unor exemple din lumea reală, care să sugereze valori care depind simultan de două repere, modele vizuale bidimensionale. Activitățile practice și exercițiile de transformare între structuri ajută elevii să înțeleagă logica organizării datelor și să dezvolte gândirea algoritmică. Se recomandă utilizarea metodelor investigative și a cooperării pe roluri: elevii lucrează în echipe, fiecare având o sarcină (implementare, comentare, testare). Instrumentele vizuale (tabele, diagrame, editoare grafice de cod) sprijină înțelegerea relațiilor ierarhice dintre structuri.
Profesorul poate introduce conceptul de recursivitate folosind metoda analogiei („oglinda într-o oglindă”) și reprezentarea grafică a apelurilor recursive. Este utilă învățarea prin simulare, pentru a vizualiza mecanismul stivei de apeluri. Metacogniția are un rol major: elevii trebuie să descrie verbal condițiile de oprire și fluxul execuției.
Pentru metoda Divide et impera, profesorul poate să introducă ideea de descompunere a problemei prin problematizare și euristică. Se recomandă construirea treptată a conceptului prin exemple familiare (găsirea elementului maxim, căutarea unui element, sortarea unei liste). Diagramele arborescente și vizualizarea recursivității facilitează înțelegerea structurii logice a algoritmului. O strategie eficientă este metacogniția – elevii sunt invitați să explice verbal modul de împărțire și combinare a subproblemelor. La predare, se evidențiază aplicarea acestei strategii și pentru metoda căutării binare, cu eliminarea uneia dintre subproblemele generate, având în vedere că nu influențează rezultatul. Se recomandă abordarea unor rezolvări cu aplicarea algoritmilor de bază (de exemplu, sume, minime), probleme clasice (de exemplu, problema turnurilor din Hanoi, problema covorului lui Sierpinski) ca model, apoi realizarea unor probleme asemănătoare care presupun adaptări, reinterpretări, extinderi.
Pentru metoda Greedy, profesorul poate utiliza învățarea prin descoperire controlată, solicitând elevilor să deducă regula de selecție optimă din exemple concrete. Este utilă strategia exemplu–contraexemplu, pentru a arăta că metoda nu este universal aplicabilă. Prin discuții metacognitive, elevii învață să argumenteze alegerile algoritmice și să compare abordări. Având în vedere că „algoritmii Greedy se folosesc de obicei în probleme de optimizare în care trebuie făcute un număr de alegeri pentru a ajunge la o soluție optimă” (Cormen, T.H., Leiserson, C.E., Rivest, R.R., *Introducere în algoritmi*), se pune accentul pe probleme de determinare a unor valori optime. Se recomandă abordarea unor rezolvări cu aplicarea algoritmilor de bază (de exemplu, sume, minime), probleme clasice (de exemplu, problema spectacolelor, problema fixării unor scânduri cu număr minim de cuie, problema rucsacului în forma continuă) ca model, apoi realizarea unor probleme asemănătoare care presupun adaptări, reinterpretări, extinderi.

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

Pentru Python:

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

Pentru C++:

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

Pentru Python și C++:

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

Instrumente online pentru implementarea soluțiilor

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

Interpretoare Python

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

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

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

Pentru obținerea unor mijloace de învățământ:

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

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

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

GRUP DE LUCRU

Nume și prenume	Grad didactic/Titlu științific	Instituție de apartenență, localitate, județ
CRĂCIUNESCU Georgeta Antonia Rodica	consilier	Ministerul Educației și Cercetării
ACIOBĂNIȚEI Iulian	conferențiar universitar, doctor	Academia Tehnică Militară „Ferdinand I”, București
ANTON Cristina Elena	profesor, gradul didactic I	Colegiul Național „Gh. M. Murgoci”, Brăila, județul Brăila
BOCA Alina Gabriela	profesor, gradul didactic I	Colegiul Național de Informatică „Tudor Vianu”, București
BORZA Diana-Laura	lector universitar, doctor	Universitatea Babeș Bolyai, Cluj-Napoca, județul Cluj
CÂRSTEA Claudia	conferențiar universitar, doctor	Academia Forțelor Aeriene „Henri Coandă”, Brașov, județul Brașov
CONTRAȘ Diana	profesor, gradul didactic I	Colegiul Național „Gheorghe Șincai”, Baia Mare, județul Maramureș
DIOSAN Laura-Silvia	profesor universitar, doctor	Universitatea Babeș-Bolyai, Cluj-Napoca, județul Cluj
DRAGOMIR-CONSTANTIN Florentina-Loredana	conferențiar universitar, doctor	Universitatea Națională de Apărare Carol I, București
DUMITRAN Adrian-Marius	lector universitar, doctor	Universitatea București, Facultatea de Matematică și Informatică
FLOREA Andrei	profesor, gradul didactic I	Colegiul Național „Ion Luca Caragiale”, București
IFTENE Adrian	profesor universitar, doctor	Universitatea „Alexandru Ioan Cuza”, Facultatea de Informatică, Iași, județul Iași
IORDAICHE Eugenia-Cristiana	profesor, gradul didactic I	Liceul Teoretic „Grigore Moisil”, Timișoara, județul Timiș
MARCU ALEXANDRA	profesor, gradul didactic I	Colegiul Naţional 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 Naţional Militar „Ştefan 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