Anexa nr. …. la ordinul ministrului educației și cercetării nr. …… /…….....

Programa școlară
pentru disciplina

INFORMATICĂ

Clasa a IX-a
Curriculum de specialitate (CS)
pentru filiera teoretică, profilul real, specializarea științe ale naturii

- 2025 –

NOTĂ DE PREZENTARE

Statutul disciplinei

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

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

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

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

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

Activitățile practice sunt desfășurate obligatoriu în laboratorul de informatică.

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

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

- Legea învățământului preuniversitar nr. 198/2023, cu modificările și completările ulterioare, precum și alte acte subsecvente relevante privind implementarea curriculumului național;
- Ordinul privind aprobarea planurilor-cadru pentru învățământul liceal cu frecvență zi (Ordinului ministrului educației și cercetării nr. 4.350/2025);
- Recomandarea Consiliului UE privind competențele-cheie pentru învățarea pe tot parcursul vieții (2018);
- Cadrul european al calificărilor (EQF);
- Rapoarte OECD/UNESCO privind competențele digitale și educația STEM, Cadre de referință;
- Profilul de formare al absolventului (Ordinul ministrului educației 6731/2023);
- Cadrul european al competențelor digitale pentru cetățeni (DigComp), respectiv Cadrul de Competențe digitale pentru elevi DigiComp 2.2. (Anexa_OM_6466_2024).

Rolul disciplinei în formarea elevilor

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

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

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

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

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

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

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

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

Categorii de programe școlare pentru disciplina *informatică*

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

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 la filiera teoretică, profilul real, specializarea matematică - informatică, și specializarea științe ale naturii, precum și la filiera vocațională, profilul militar, specializarea matematică - informatică militară;
- utilizarea C++, suplimentar, ca al doilea limbaj de programare, doar la filiera teoretică, profilul real, specializarea matematică-informatică, pentru clase cu predarea disciplinei *informatică* în regim intensiv;
- utilizarea SQL în clasa a XII-a, la filiera teoretică, profilul real, specializarea matematică - informatică și la filiera vocațională, profilul militar, specializarea matematică - informatică militară;
- utilizarea unor resurse digitale moderne în laboratoare de informatică dotate adecvat;
- corelarea activităților de învățare cu domenii STEM, economie, științe sociale și arte digitale.

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

Alegerea limbajului 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.

Pentru filiera teoretică, profilul real, specializarea matematică -informatică, la clasele cu predarea disciplinei *informatică* în regim intensiv, studierea limbajului C++ se justifică prin valoarea formativă și complexitatea acestuia, care 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 de *informatică* este esențială 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 (CG, CS, conținuturi, exemple de activități de învățare) în modalități și mijloace pentru realizarea demersului didactic, prin exemple minimale, relevante, de abordare a activității didactice, pentru alegerea strategiilor didactice și pentru integrarea conținuturilor și competențelor în practica școlară.

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

COMPETENȚE GENERALE (CG)

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

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

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

Clasa a IX-a
CS 1.1. Identifică principalele caracteristici ale organizării datelor în cadrul unor modele conceptuale fundamentale - date simple sau liste, pentru a structura și accesa date în vederea prelucrării acestora
- *recunoașterea caracteristicilor unei liste evidențiind modul în care se succed mașinile care sunt oprite la semafor pe o stradă cu sens unic, faptul că fiecare mașină, cu excepția primei și ultimei mașini, are imediat înaintea ei și imediat după ea câte o altă mașină* - *enumerarea reperelor pentru parcurgerea unei liste în vederea identificării perechilor de elemente aflate pe poziții consecutive și care au aceleași caracteristici* - *recunoașterea caracteristicilor unei liste cu acces direct prin nominalizarea celui de al cincilea elev din catalog, fără a fi necesară parcurgerea tuturor datelor elevilor care îl preced în catalog*
CS 1.2. Identifică specificul, caracteristicile și etapele unor algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea algoritmică a numerelor, sortarea sau generarea sistematică a unor secvențe de valori
- *enumerarea etapelor algoritmului lui Euclid pentru determinarea celui mai mare divizor comun cu scopul determinării dimensiunii maxime a unei plăci de gresie de formă pătrată folosită pentru acoperirea completă, cu plăci întregi, a podelei unei săli de dimensiuni date* - *reamintirea secvenței de operații necesare pentru a adăuga o cifră la dreapta celorlalte cifre ale unui număr, în contextul determinării valorii rezultate în urma eliminării tuturor cifrelor impare ale unui număr, evidențiind necesitatea păstrării ordinii inițiale a cifrelor pare rămase și a cifrelor nule de la finalul numărului* - *reamintirea regulii de obținere a unui termen al șirului Fibonacci, pe baza celor doi termeni anteriori, în contextul unui proiect de cercetare având ca temă șirul lui Fibonacci în natură* - *identificarea caracteristicilor metodei de sortare prin selecția minimului aplicată scenariului de aranjare a finaliștilor unui concurs de talente în ordinea crescătoarea a scorului obținut*
CS 1.3. Identifică specificul, caracteristicile și etapele unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor
- *identificarea datelor cu care lucrează algoritmii, pe baza caracteristicilor acestora (date de intrare, date ieșire, date de manevră)* - *conceperea unei diagrame care ilustrează etapele de elaborare a unui program* - *enumerarea etapelor de elaborare a unui program, precizând rolul fiecăreia* - *asocierea unor termeni precum modul, gândire computațională, proiectare cu rolul lor în rezolvarea unei probleme informatice*
CS 1.4. Identifică principalele elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program, în vederea rezolvării eficiente a problemelor informatice
- *recunoașterea operatorilor specifici clasei list din Python ([], in, not in, +, *, ==, !=, <, >), în contextul recapitulării modalităților de lucru cu structuri de date, prin completarea unui tabel care asociază fiecare operator cu rolul său* - *descrierea unor metode suplimentare ale clasei list din Python (extend(), reverse(), clear(), remove(), sort(), copy()), în contextul recapitulării instrumentelor de prelucrare a colecțiilor de date, prin completarea unui tabel care asociază fiecare metodă cu acțiunea sa principală asupra listei* - *enumerarea etapelor în lucrul cu un fișier text și a metodelor corespunzătoare în Python: open(), read(), write() și close()* - *identificarea pe un exemplu de program a subprogramelor asociate butoanelor sau evenimentelor de tip "apăsarea unei taste"*
CS 1.5. Identifică elementele de sintaxă și componentele din definiția și apelul subprogramelor și subprogramelor predefinite pentru operații uzuale, pentru a le utiliza în implementarea algoritmilor în limbaj de programare
- *descrierea structurii de bază a unui subprogram în Python (definire cu def, parametri, instrucțiuni, apel) în contextul recapitulării modului de organizare a codului într-un proiect existent, prin completarea unui șablon de cod parțial* - *recunoașterea funcțiilor predefinite Python pentru calcule matematice uzuale ((abs(), round(), int(), sqrt()), într-un program dat, de rezolvare a unei probleme de calcul numeric* - *identificarea funcțiilor predefinite Python utilizate pentru colecții (len(), min(), max(), sum()), în contextul recapitulării elementelor de bază privind manipularea listelor numerice, prin completarea unui tabel care asociază fiecare funcție cu rezultatul obținut pentru o listă dată*

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

Clasa a IX-a
CS 2.1. Explică principiile de organizare a datelor în cadrul unor modele conceptuale fundamentale - date simple sau liste, pentru a le gestiona eficient
- *explicarea rolului unei liste de frecvențe în organizarea și interpretarea datelor dintr-o situație reală, exemplificată printr-o listă care centralizează voturile obținute de mai mulți candidați, interpretând semnificația valorilor înregistrate în listă ca frecvențe ale opțiunilor exprimate* - *explicarea modului în care parcurgerea secvențială a datelor permite extragerea de informații esențiale dintr-o colecție organizată liniar, exemplificând prin numărarea persoanelor care au participat la un eveniment, pe baza unui criteriu dat* - *explicarea necesității memorării datelor într-o listă pentru a permite prelucrări ulterioare, exemplificând prin înregistrarea zilnică a numărului de clienți ai unui magazin într-o lună pentru a identifica zilele cu numărul minim de clienți*
CS 2.2. Explică etapele algoritmilor specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea algoritmică a numerelor, sortarea sau generarea sistematică a unor secvențe de valori
- *explicarea procesului de formare a unui număr nou prin adăugarea repetată a câte unei cifre în stânga celor existente, evidențiind înmulțirea cifrei adăugate cu o variabilă, pe baza unui algoritm dat* - *explicarea modului de formare a unei liste de valori ai cărei termeni respectă o anumită regulă (de exemplu, lista temperaturilor medii anuale în procesul de încălzire globală știind temperaturile din ani consecutivi)* - *explicarea aplicabilității proporției de aur în știință, artă și în natură, pe baza materialelor de documentare primite, privind șirul lui Fibonacci* - *explicarea procesului de aranjare în ordine crescătoare a șapte comenzi online în funcție de valoarea comenzilor, folosind selecția minimului, cu reprezentarea ordinii comenzilor după fiecare parcurgere*
CS 2.3. Explică etapele unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor
- *explicarea etapelor a două strategii de rezolvare prin proiectarea modulară a algoritmilor pentru aceeași problemă, subliniind diferențele de organizare a modulelor și impactul acestora asupra clarității și eficienței soluției* - *explicarea rolului și importanței fiecărei etape din procesul de dezvoltare a unui program* - *explicarea rolului fiecărui modul în cadrul unei aplicații cunoscute, pentru realizarea obiectivelor acesteia*
CS 2.4. Explică rolul principalelor elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program în vederea rezolvării eficiente a problemelor informatice
- *explicarea modului în care caracteristicile clasei list (mutabilitate, acces prin index, ordinea inserării) influențează manipularea datelor, în contextul unei aplicații de gestionare a notelor elevilor, prin descrierea pașilor de adăugare, modificare și ștergere a elementelor* - *explicarea modului de funcționare a metodelor append(), insert() și pop() în contextul unei aplicații de gestionare a unei liste de participanți la un eveniment, prin justificarea efectelor fiecărei metode asupra structurii listei* - *explicarea mecanismelor de citire și scriere într-un fișier și a noțiunii de "sfârșit de fișier"* - *explicarea rolului principalelor obiecte grafice într-o aplicație mai complexă, dată, care utilizează cât mai multe obiecte grafice (fereastră, etichetă, buton, casetă de text, listbox, messagebox, meniu)*
CS 2.5. Explică mecanismul de executare a subprogramelor și subprogramelor predefinite pentru operații uzuale, pentru a le utiliza în implementarea algoritmilor în limbaj de programare
- *explicarea rolului subprogramelor într-un proiect de organizare modulară a unui algoritm dat, care determină câte numere dintr-un șir de valori au suma cifrelor un număr prim, prin descrierea relației dintre antet, parametri și corpul fiecărui subprogram* - *explicarea modului de funcționare a apelului unui subprogram în contextul unei aplicații de calcul al costului total al cumpărăturilor, prin descrierea legăturii dintre parametri, valoarea returnată și afișarea rezultatului* - *explicarea modului de funcționare a subprogramelor predefinite pentru conversii și rotunjiri, în contextul unui exemplu practic de prelucrare a notelor școlare, prin precizarea diferenței dintre int() și round()* - *explicarea modului de funcționare a programelor ce utilizează funcțiile len(), min(), max() și sum() în contextul unei aplicații de gestiune a notelor elevilor, prin argumentarea legăturii dintre fiecare funcție și informația pe care o furnizează despre colecție*

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

Clasa a IX-a
CS 3.1. Utilizează modul de organizare și prelucrare a datelor în cadrul unor modele conceptuale fundamentale - date simple sau liste, pentru a rezolva probleme de gestionare a datelor

Clasa a IX-a
- *aplicarea conceptului de listă de frecvențe în analiza răspunsurilor dintr-un chestionar de satisfacție, prin numărarea aparițiilor fiecărei variante de răspuns (1–5 stele) și organizarea rezultatelor sub formă de listă pentru identificarea tendințelor generale* - *aplicarea principiilor parcurgerii liniare pentru folosirea valorilor care respectă o anumită condiție într-un context real, cum ar fi verificarea listelor de plăți lunare pentru determinarea facturilor restante, prin procesarea secvențială a fiecărui element fără memorarea rezultatelor intermediare* - *aplicarea conceptului de parcurgere liniară cu memorare prin stocarea notelor obținute de elevii dintr-o clasă la un test, utilizând lista pentru determinarea elevilor care au notele mai mari decât media clasei*
CS 3.2. Aplică algoritmi specializați pe clase de probleme, pentru a îi putea utiliza în prelucrarea algoritmică a numerelor, sortarea sau generarea sistematică a unor secvențe de valori
- *simularea procesului de obținere a divizorilor unui număr natural dat utilizând reperele pentru parcurgerea acestora* - *implementarea unor algoritmi de divizibilitate dați pentru a verifica dacă un număr natural este perfect, determinând suma divizorilor acestuia* - *reprezentarea în pseudocod a algoritmului lui Euclid pentru determinarea celui mare divizor comun a două numere naturale nenule citite, pentru utilizarea acestuia în contextul simplificării unei fracții* - *aplicarea algoritmului de generare a elementelor unei liste ai cărei termeni respectă o regulă dată, în contextul determinării numărului de ani după care soldul contului bancar, având o sumă inițială și o dobândă anuală, depășește o valoare dată* - *implementarea algoritmului de sortare cu listă de frecvențe pentru ordonarea notelor la testul de la informatică al unei clase de elevi (note întregi între 1 și 10), cu construirea listei de frecvențe prin numărarea frecvenței fiecărei note și afișarea notelor sortate prin parcurgerea listei*
CS 3.3. Aplică strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor
- *descompunerea unei probleme, având descrierea rezolvării acesteia, în subprobleme (module), indicând etapele de lucru pentru fiecare* - *utilizarea principiilor de proiectare modulară pentru o aplicație simplă (de exemplu, gestionarea unor cheltuieli și încasări lunare), atribuind câte un modul fiecărui membru al unei echipe* - *aplicarea unui model de proiectare modulară în realizarea unei soluții, în vederea efectuării etapelor de analiză, proiectare, testare în cadrul fiecărui modul*
CS 3.4. Utilizează principalele elemente ale limbajului de programare utilizate pentru prelucrarea datelor organizate în modele fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program, în vederea rezolvării eficiente a problemelor informatice
- *aplicarea operatorilor de concatenare (+) și multiplicare (*) în rezolvarea unei situații practice de gestionare a comenzilor din două depozite, prin combinarea listelor de produse și dublarea unui stoc existent* - *aplicarea metodelor count() și index() în rezolvarea unei situații practice de numărare a produselor identice și de identificare a poziției unui articol într-o listă de stocuri, prin scrierea unui program Python simplu* - *aplicarea metodelor remove() și clear() pentru prelucrarea datelor într-o aplicație de gestionare a unui coș de cumpărături, prin scrierea unui program Python simplu care elimină produsele selectate de utilizator și golește lista după finalizarea comenzii* - *completarea unui program "lacunar" care citește o valoare întreagă dintr-un fișier text pentru a scrie diverse rezultate obținute tot într-un fișier text (de exemplu pătratul și rădăcina pătrată sau divizorii acestuia)* - *adăugarea unei casete de text într-o aplicație simplă, dată, cu o fereastră, o casetă de text și un buton care afișează un rezultat într-o fereastră de mesaje - MessageBox (de exemplu numărul de divizori), astfel încât rezultatul afișat să reflecte ambele valori din casetele de text (de exemplu numărul de divizori comuni)*
CS 3.5. Utilizează elementele de sintaxă și componentele din definiția și apelul subprogramelor și subprogramelor predefinite pentru operații uzuale, în implementarea algoritmilor în limbaj de programare
- *folosirea conceptului de transmitere a parametrilor în rezolvarea unei situații practice de calcul al temperaturii maxime, prin apelul unui subprogram Python ce returnează maximul dintre două numere* - *implementarea sintaxei de definire și apel a unui subprogram Python pentru rezolvarea unei situații practice de calcul al ariei unui teren, folosind ca parametri lungimea și lățimea introduse de utilizator* - *implementarea subprogramelor cu funcțiile predefinite abs() și sqrt() în rezolvarea unei situații practice de calcul al distanței dintre două puncte într-un plan, folosind ca parametri coordonatele introduse de utilizator* - *aplicarea funcțiilor predefinite pentru colecții în rezolvarea unei situații practice de calcul al performanței sportivilor (de exemplu, determinarea mediei, a celei mai mari și celei mai mici valori a scorurilor), prin scrierea unui program Python simplu*

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

Clasa a IX-a
CS 4.1. Analizează caracteristicile, modul de prelucrare, avantajele și dezavantajele utilizării unor modele conceptuale fundamentale - date simple sau liste, pentru a alege structura adecvată în rezolvarea unor probleme concrete
- *analizarea modului de formare și interpretare a unei liste de frecvențe în studierea obiceiurilor de cumpărare ale clienților, prin descompunerea procesului în etape (colectare, numărare, reprezentare), corelarea valorilor din listă cu comportamentele observate* - *analizarea modului de funcționare a algoritmilor de parcurgere liniară prin compararea etapelor de verificare, selecție și filtrare a datelor într-o situație practică precum monitorizarea tranzacțiilor zilnice ale unui magazin, pentru a evidenția ordinea logică a operațiilor și dependențele dintre ele* - *analizarea modului de organizare și prelucrare a unei liste de valori memorate în monitorizarea zilnică a consumului de apă al unei gospodării, prin identificarea relației dintre ordinea datelor, valorile stocate și calculele derivate, precum determinarea consumului mediu și a variațiilor semnificative*
CS 4.2. Analizează comportamentul, aplicabilitatea, avantajele și dezavantajele utilizării unor algoritmi specializați pe clase de probleme pentru prelucrarea numerelor, sortarea sau generarea sistematică a unor secvențe de valori
- *analizarea a doi algoritmi alternativi dați pentru determinarea divizorilor unui număr în vederea estimării numărului de operații efectuate* - *urmărirea raționamentului rezultat prin operațiile din cadrul unui algoritm pentru a verifica dacă acesta determină obținerea oglinditului unui număr natural* - *analizarea avantajelor utilizării divizorilor unui număr în contexte practic-aplicative (de exemplu, găsirea tuturor posibilităților pentru tăierea unei benzi de L centimetri în bucăți de lungimi numere naturale egale, fără pierderi de material, posibilitatea de distribuire a N caiete unui număr de elevi astfel încât aceștia să primească același număr de caiete, ambalarea a N produse în cel puțin două pachete astfel încât numărul de pachete să fie minim și în fiecare pachet să fie același număr de produse)* - *analizarea comportamentului unui algoritm pentru formarea unei liste date (de exemplu, lista primelor n pătrate perfecte consecutive), urmărind raționamentul rezultat prin operațiile din cadrul acestuia pentru a aprecia adecvarea acestuia la cerință* - *analizarea algoritmului de sortare prin selecția minimului, pentru o lista cu n valori, estimând numărul de comparări pentru un caz favorabil, un caz mediu și un caz nefavorabil*
CS 4.3. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor
- *analizarea erorilor apărute în programele informatice în scopul încadrării lor într-una dintre categoriile: erori de sintaxă, de logică, de executare* - *analizarea modulelor unui cod sursă al unui sistem complex de validare a datelor (de ex., un sistem care validează formulare web), punând în evidență contribuția fiecărui modul la procesul general de validare* - *analizarea avantajelor și dezavantajelor utilizării modulelor în cadrul elaborării unui program* - *analizarea modului în care interacționează modulele unei aplicații, pe baza unei diagrame care arată componentele acesteia și modul în care interacționează*
CS 4.4. Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor elemente ale limbajului de programare, pentru a identifica soluția adecvată prelucrării datelor organizate în modele fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program, în vederea rezolvării eficiente a problemelor informatice
- *analizarea diferențelor de implementare în Python între atribuirea directă a listelor și copierea acestora prin metode specifice (copy()-copiere superficială, slicing-selectarea unei secvențe), în contextul unei aplicații de procesare a datelor financiare, prin observarea efectelor modificării elementelor asupra listelor asociate* - *analizarea efectelor utilizării metodelor copy() și sort() asupra conținutului listelor, în contextul unei aplicații de evidență a notelor elevilor, prin compararea rezultatelor obținute înainte și după sortare, respectiv copiere* - *analizarea diferențelor de comportament dintre metodele sort() și reverse() din Python aplicate asupra unei liste de prețuri, în contextul unei aplicații de evidență a vânzărilor, prin observarea și explicarea ordinii obținute și a modificării listei inițiale* - *compararea timpului de executare a unui program care citește datele dintr-un fișier în raport cu același program care citește datele de la tastatură* - *analizează avantajele și dezavantajele organizării elementelor grafice într-o fereastră, a textelor și a mesajelor asociate, în logica "user-friendly=ușor de utilizat"*
CS 4.5. Analizează aplicabilitatea, avantajele și dezavantajele utilizării subprogramelor, în implementarea algoritmilor în limbaj de programare
- *analizarea descompunerii unui program complex de prelucrare a datelor în subprograme, prin identificarea variabilelor locale și globale și explicarea modului în care acestea influențează funcționarea independentă a fiecărui subprogram* - *analizarea diferențelor dintre apelul unui subprogram cu parametri și fără parametri, în contextul unui proiect de prelucrare a datelor meteo, prin observarea modului în care se transmit și se utilizează datele*

Clasa a IX-a
- *analizarea efectului diferitelor funcții de conversie (int(), float(), str()) asupra acelorași date de intrare, în contextul unei aplicații de procesare a prețurilor produselor folosind subprograme, prin compararea rezultatelor și explicarea diferențelor de tip de date* - *analizarea modului în care rezultatele funcțiilor min() și max() pot fi influențate de structura și tipul elementelor din colecție, în contextul unei aplicații de prelucrare a prețurilor produselor folosind subprograme, prin compararea rezultatelor pentru liste eterogene*

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

Clasa a IX-a
CS 5.1. Argumentează alegerea unor modele conceptuale fundamentale - date simple sau liste - și a modurilor de prelucrare a acestora pentru rezolvarea unor probleme informatice concrete
- *evaluarea relevanței și acurateței unei liste de frecvențe în prezentarea rezultatelor unei anchete de opinie, prin verificarea modului în care datele reflectă realitatea studiată, argumentarea corectitudinii structurii listei și formularea unei concluzii privind utilitatea sa pentru luarea deciziilor* - *alegerea unui algoritm de parcurgere liniară fără memorare pentru procesarea datelor provenite dintr-un registru de intrări, prin aprecierea clarității pașilor logici, a modului de acces la date și argumentarea deciziei privind eficiența și corectitudinea abordării alese* - *evaluarea corectitudinii și relevanței memorării datelor într-o listă utilizată pentru înregistrarea și compararea vânzărilor lunare ale unui produs, prin verificarea completitudinii datelor stocate, a ordinii lor și a modului de interpretare în raport cu scopul analizei comerciale*
CS 5.2. Evaluează corectitudinea și eficiența soluțiilor cu algoritmi specializați pe clase de probleme pentru prelucrarea numerelor, sortarea sau generarea sistematică a unor secvențe de valori
- *evaluarea complexității a doi algoritmi dați pentru determinarea divizorilor unui număr natural stabilind care dintre cei doi algoritmi este mai eficient (de exemplu, parcurgere a posibililor divizori de la 1 până la n, până la n/2 sau până la $\sqrt{n}$)* - *argumentarea faptului că un algoritm ce construiește oglinditul unui număr și apoi oglinditul oglinditului nu conduce totdeauna la obținerea numărului inițial, susținută de teste adecvate și compararea rezultatelor obținute cu cele așteptate* - *justificarea folosirii unui algoritm de generare recurentă a termenilor unei expresii matematice pentru însumarea unor produse, în comparație cu folosirea unui algoritm de calcul direct al fiecărui termen al acesteia (de exemplu pentru a calcula suma $1+1\cdot2+1\cdot2\cdot3+...+1\cdot2\cdot3\cdot...\cdot n$)* - *evaluarea eficienței (calculând ordinul de complexitate) unui algoritm de obținere a termenilor șirului lui Fibonacci, pornind de la un termen dat, în ordine descrescătoare, prin două metode: prima metodă memorează ultimul termen afișat apoi îl determină pe cel curent, care îl precede în șir, prin generarea, începând de la 1, în ordine crescătoare, a tuturor termenilor, până la cel căutat, iar a doua metodă generează toți termenii în ordine crescătoare, o singură dată, începând de la 1, până la termenul care îl precede pe cel dat inițial, apoi generarea fiecărui termen curent se face prin diferență, pe baza adaptării regulii de generare specifice* - *evaluarea corectitudinii unui algoritm care generează o listă cu primii termeni ai șirului lui Fibonacci, pe baza unor teste adecvate care au în vedere și cazuri limită, de exemplu listă vidă, listă cu un singur element, listă cu doar două elemente* - *argumentarea deciziei de utilizare a metodei de sortare cu listă de frecvențe pentru un sistem de triaj al pacienților din camera de urgență a unui spital, sistem bazat pe ordonarea descrescătoare a gravității stării lor medicale (codificată prin valori de la 1 la 5), ținând cont de caracteristicile datelor (număr mare de elemente, interval numeric limitat)*
CS 5.3. Evaluează corectitudinea și eficiența algoritmilor care utilizează strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor
- *argumentarea avantajelor descompunerii unei probleme complexe în etape sau subprobleme mai simple (citire date, validare, rezolvare propriu-zisă, afișare rezultat) din perspectiva verificării corectitudinii fiecărui modul* - *evaluarea timpului de executare a diferiților algoritmi pentru seturi de date de dimensiuni diferite și reprezentarea grafică a rezultatelor* - *evaluarea comparativă a două soluții ale unei probleme, una modulară și una compactă, argumentând care este mai eficientă* - *evaluarea corectitudinii și eficienței unei soluții propuse de colegi, oferind feedback privind claritatea și independența modulelor*
CS 5.4. Evaluează corectitudinea și eficiența aplicațiilor care utilizează elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program, în vederea rezolvării eficiente a problemelor informatice
- *justificarea corectitudinii și eficienței utilizării operatorilor de apartenență (in, not in) într-un program de validare a datelor introduse de utilizator, în cadrul unei dezbateri despre criterii de claritate și siguranță în prelucrarea listelor* - *evaluarea unui program care utilizează obiecte din clasa list, și determină numărul de perechi de valori pare, aflate pe poziții consecutive în listă, pe baza unor teste adecvate care au în vedere comportamentul acestuia pentru elemente care au câte doi vecini, sau se află la fiecare dintre cele două extremități ale listei*

Clasa a IX-a
- *evaluarea corectitudinii unui program cu fișiere text, pe baza unor teste adecvate care au în vedere și cazuri limită, de exemplu fișier inexistent, fișier gol* - *compararea unei aplicații cu interfață grafică în raport cu aceeași aplicație cu interfață text, evidențiind valențele fiecăreia (avantaje și dezavantaje)*
CS 5.5. Evaluează corectitudinea și eficiența programelor care utilizează subprograme, pentru a rezolva probleme informatice
- *susținerea cu argumente a celor două variante de implementare a aceluiași algoritm (una cu subprograme, alta fără), în cadrul unei dezbateri despre eficiența, claritatea și reutilizarea codului* - *justificarea alegerii unei variante de definire și apel al unui subprogram Python, din mai multe variante posibile, într-un proiect de calcul al salariului net, prin aplicarea unor criterii de lizibilitate, modularitate și reutilizare a codului* - *evaluarea acurateței și relevanței utilizării funcțiilor de rotunjire (round(), math.floor(), math.ceil()) într-o aplicație de calcul al totalului facturii, în contextul unei discuții despre criterii de precizie financiară* - *justificarea utilizării funcției sum() pentru calculul sumei elementelor dintr-o colecție, în comparație cu iterarea explicită, evidențiind avantajele din perspectiva clarității codului și reducerii erorilor, în contextul unei discuții despre bune practici de programare și optimizare a codului* - *evaluarea corectitudinii unei aplicații cu subprograme, pe baza unor teste adecvate care au în vedere și cazuri limită, de exemplu pentru parametri cu valori inadecvate, lipsa unei valori returnate pentru unele cazuri*

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

Clasa a IX-a
CS 6.1. Proiectează algoritmi personalizați care utilizează modele conceptuale fundamentale - date simple sau liste pentru organizarea și prelucrarea eficientă a datelor, utilizând algoritmi de bază, în vederea rezolvării unor probleme
- *proiectarea unui model conceptual de listă de frecvențe pentru monitorizarea numărului de erori raportate zilnic, pentru fiecare tip posibil, într-un centru de asistență tehnică, prin stabilirea structurii listei, a modului de înregistrare a fiecărei erori și a regulilor de actualizare a frecvențelor, astfel încât modelul să poată fi ulterior implementat în limbaj de programare* - *proiectarea unui algoritm de parcurgere liniară pentru determinarea și prelucrarea automată a valorilor neconforme dintr-o listă de date financiare, prin stabilirea pașilor de acces succesiv la elemente, definirea condițiilor de verificare și formularea logicii generale de procesare, care va constitui ulterior baza implementării în limbaj de programare* - *proiectarea unei liste de valori memorate pentru urmărirea evoluției zilnice a nivelului de poluare într-un oraș, prin definirea modului de colectare și stocare a valorilor unui anumit tip de noxe, stabilirea algoritmilor de parcurgere pentru calculul valorii medii și identificarea zilelor critice cu valori care depășesc media, ca bază pentru o viitoare implementare în limbaj de programare*
CS 6.2. Proiectează soluții cu aplicarea personalizată a algoritmilor specializați pe clase de probleme pentru prelucrarea numerelor, sortarea sau generarea sistematică a unor secvențe de valori
- *proiectarea unui program ce implementează operații cu câte două fracții date prin numărătorul și numitorul lor (simplificare, adunare, scădere, înmulțire, împărțire), utilizând personalizat algoritmi de divizibilitate* - *rezolvarea unei probleme ce determină cel mai mic multiplu comun a două numere pe baza celui mai mare divizor comun al lor, pentru obținerea duratei minime de timp după care două semafoare trec simultan la aceeași culoare, cunoscându-se, pentru fiecare, timpul de menținere a fiecărei culori (roșu și verde)* - *conceperea unui algoritm care determină numărul de persoane care sunt născute într-o anumită lună a anului pe baza unei liste de coduri numerice personale date* - *crearea unui algoritm eficient care afișează primii n termeni ai unui șir recurent, prin determinarea formulei termenului general, cunoscându-se primii termeni și relația de recurență (de exemplu, pentru $a_0=3$ și relația de recurență $a_n=2\cdot a_{n-1}$, pentru $n>0$, se obține $a_n=3\cdot 2^n$)* - *implementarea unui program pentru gestionarea solicitărilor de mentenanță pentru o companie de lifturi, care își sortează cererile în funcție de tipul defecțiunii (codificat cu valori de la 1 la 8), obținându-se în final lista cererilor ordonate*
CS 6.3. Proiectează modular algoritmi personalizați pentru rezolvarea problemelor
- *descompunerea unei aplicații în module, în cadrul unui grup, în care fiecare modul este abordat de câte un membru al grupului, și integrarea acestor rezolvări, ulterior, într-un produs unitar* - *conceperea unei diagrame generale pentru rezolvarea modulară a unei probleme, evidențiind componentele și conexiunile dintre ele* - *completarea unei aplicații existente (de exemplu: o aplicație de gestionare a bibliotecii școlare, un sistem de evidență a prezenței, un registru digital etc.), a cărei descriere este dată, cu un modul nou, care adaugă o funcționalitate suplimentară (de exemplu, raportare, notificări, export de date, interfață de administrare etc.), respectă principiile de proiectare modulară și poate fi integrat în structura existentă*

Clasa a IX-a
CS 6.4. Implementează aplicații care integrează în mod personalizat elemente specifice de limbaj de programare pentru prelucrarea datelor organizate în modele fundamentale - date simple sau liste, respectiv pentru transmiterea datelor de la și către un program, în vederea realizării unor soluții funcționale și performante
- *elaborarea unei aplicații Python de gestiune a listelor de produse dintr-un magazin online, prin integrarea operatorilor de acces, apartenență, concatenare și relaționare pentru actualizarea și compararea automată a stocurilor din mai multe surse* - *proiectarea unei aplicații Python pentru administrarea unei liste de produse dintr-un magazin, utilizând metodele clasei list (append(), insert(), pop(), count(), index(), copy(), sort()) pentru a adăuga, elimina, sorta și analiza elementele din stoc* - *elaborarea unei aplicații Python pentru gestionarea unei liste dinamice de sarcini (to-do list), prin identificarea și utilizarea metodelor adecvate (append(), remove(), sort(), reverse(), copy(), clear()) pentru adăugarea, eliminarea, ordonarea și resetarea elementelor* - *proiectarea unei aplicații de exploatare cu valențe practice a unui fișier text (de generare a unor serii aleatoare de numere folosind biblioteca random urmată de numărări și rezultate statistice pentru o astfel de serie generată)* - *implementarea unei aplicații Python pentru afișarea, într-o fereastră, cu fundal colorat, a câte unui termen aparținând unui șir generat pe baza unei relații de recurență, utilizând două obiecte din clasa Button, cu proprietatea text completată sugestiv, pentru a marca pornirea, respectiv oprirea procesului de obținere a termenilor șirului*
CS 6.5. Implementează aplicații în limbaj de programare care integrează subprograme personalizate, pentru a modulariza și organiza eficient codul în cadrul soluțiilor informatice
- *construirea unui mini-proiect de tip aplicație modulară Python (de exemplu, gestionarea unui catalog școlar), prin proiectarea și implementarea propriilor subprograme care utilizează parametri și returnează valori, cu prezentarea funcționalității în fața clasei* - *proiectarea unei aplicații modulare Python de tip calculator personalizat (ex. conversii valutare, conversii de unități), prin definirea și apelarea mai multor subprograme proprii, utilizând parametri și valori returnate relevante pentru funcționalitatea dorită* - *construirea unui mini-proiect de tip aplicație modulară Python pentru conversia și prelucrarea valorilor numerice (de exemplu, o aplicație care convertește temperaturi, distanțe sau sume de bani), prin integrarea mai multor funcții predefinite de calcul și conversie (abs(), round(), int(), float(), sqrt())* - *proiectarea unei aplicații modulare Python care gestionează o colecție de date reale (de exemplu, temperaturi zilnice, vânzări lunare, note școlare), utilizând funcțiile len(), min(), max(), sum() pentru generarea unui raport sintetic cu informații despre datele analizate*

CONȚINUTURI ALE ÎNVĂȚĂRII

Clasa a IX-a

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

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

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, academic, cât și practic, 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, de regulă, 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 aplicațiile specifice, iar organizarea calculatoarelor 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 elevilor din clasă/școala în care predă profesorul.

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

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.

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.
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.
Mijloacele de învățământ utilizate pot fi variate, beneficiind de tehnologiile moderne care facilitează învățarea, cum ar fi aplicații specializate, software-uri educaționale, simulatoare ale comportamentului datelor prelucrate de algoritmi, tutoriale, resurse online, platforme care permit evaluarea automată a soluțiilor informatice.
Evaluarea în cadrul disciplinei informatică trebuie să aibă un caracter formativ/pe parcurs, urmărind nu doar obținerea unui rezultat corect, ci și modul în care elevii își formează gândirea algoritmică, formulează strategii, își testează algoritmii și corectează propriile erori. Se recomandă evaluări sumative/finale, după fiecare unitate de învățare.
Conform sugestiilor metodologice din programa școlară de *informatică și TIC* pentru gimnaziu, la clasele a VII-a și a VIII-a pentru formarea competențelor specifice s-a putut utiliza un limbaj de programare dintr-o listă care cuprinde, de exemplu, propuneri ca Python, C, C++. Deoarece alegerea limbajului de programare a fost la latitudinea profesorului, este necesară **o armonizare a acestor limbaje de programare la trecerea în clasa a IX-a, pentru ca toți elevii să utilizeze limbajul de programare precizat în programa școlară în vigoare pentru această clasă**. Astfel, se recomandă prezentarea, la început, a unor elemente de corespondență a principalelor instrumente de reprezentare a algoritmilor în pseudocod, blocuri grafice, limbaje de nivel înalt Python și C++: citire și afișare a datelor, biblioteci, comenzi și scripturi, instrucțiuni, variabile și tipuri de date de bază, operatori de bază, medii integrate pentru dezvoltarea programelor (IDE) și funcționalități ale acestora.

Programa disciplinei informatică are în vedere conținuturi grupate în domenii specifice, de exemplu:
1. Modelele conceptuale de organizare a datelor, care vizează reprezentări abstracte ale modului în care sunt structurate și relaționate datele într-un sistem informatic — înainte ca ele să fie memorate efectiv printr-un program sau o bază de date. Ele răspund la întrebarea: „Ce informații trebuie să prelucrez/stochez și cum sunt legate între ele?” și nu încă 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, asociative sau mixte, modele conceptuale complexe - relaționale, ierarhice sau asociative, 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).

Un exemplu, orientativ, de ordine de abordare a conținuturilor pentru clasa a IX-a, specializarea științe ale naturii:

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

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

de nivel și specializare), în etapa de proiectare, respectiv de implementare. Astfel, la rezolvarea **fiecărei probleme** de natură algoritmică, pe parcursul studiului disciplinei, se evidențiază aspecte specifice etapelor de elaborare a programelor:
- analiză (identificare a datelor de intrare și de ieșire, organizare conceptuală a datelor, descompunere a unei probleme în subprobleme, identificare a unor modele repetitive, abstractizare);
- proiectare (descompunere în module, reprezentare ca schemă logică, pseudocod);
- implementare (scriere a codului în limbaj de programare);
- testare (criterii de elaborare a testelor pentru validarea datelor și pentru verificarea comportamentului programului, urmărire a evoluției valorilor variabilelor pentru identificarea și tratarea eventualelor erori).

În parcurgerea conținuturilor, se recomandă rezolvarea unor probleme concrete, întâlnite în viața reală, pentru a evidenția succesiunea etapelor de elaborare a unui program și pentru a dezvolta gândirea algoritmică a elevilor. Profesorul poate alterna reprezentările grafice, textuale și codificate ale aceluiași algoritm pentru a facilita înțelegerea legăturii dintre abstractizare și implementare.

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

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

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

Programarea orientată pe obiecte se poate introduce la clasa a IX-a prin analogii intuitive cu lumea reală, evidențiind faptul că orice obiect are proprietăți (date) și comportamente (metode), pentru a facilita înțelegerea conceptelor de clasă și membri ai clasei.

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

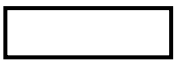
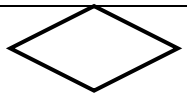
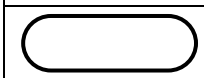
Lucrul cu fișiere text se poate aborda prin activități practice de învățare care ilustrează procesul complet de deschidere, citire, scriere și închidere a unui fișier, folosind exemple apropiate de experiența elevilor (liste de elevi, jurnale, notițe).

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

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

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

Pentru reprezentarea schemelor logice, simbolurile uzuale sunt:

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

○	Bloc conector: reprezintă conectarea a două sau mai multe puncte ale schemei logice

Pentru reprezentarea algoritmilor în pseudocod, convențiile uzuale sunt:
Indentare: liniile sunt indentate pentru a indica faptul că sunt conținute într-o structură dintr-o linie anterioară
Operațiile de intrare și ieșire sunt precizate prin comenzile **citește <identificator sau identificatori, separați prin virgulă>** respectiv **scrie <expresie sau expresii, separate prin virgulă>**
Operațiile de atribuire sunt precizate în următorul format: **<identificator> ← <expresie>**
Structurile de control sunt precizate în următoarele formate:
- structură de decizie

```
┌dacă <condiție> atunci
│  <Instrucțiuni>
│[altfel
│  <Instrucțiuni>]
└■
```

sau

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

- structură repetitivă cu test inițial

```
┌cât timp <condiție> execută
│  <Instrucțiuni>
└■
```

sau

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

- structură repetitivă cu test final

```
┌repetă
│  <Instrucțiuni>
└până când <condiție>
```

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

```
┌pentru <contor>←<expInit>,<expFin>[,<pas>] execută
│  <Instrucțiuni>
└■
```

sau

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

Subprogramele/modulele sunt definite în formatele:

```
┌subprogram <identificator> [(<parametri formali>)] [returnează <rezultate sau tipuri de date>]
│  <Instrucțiuni>
└■
```

sau

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

Un **element al unei liste indexate** se accesează precizând poziția ca indice (sau indici, separați prin virgulă), ai variabilei de tip listă: de exemplu a_i sau $a_{i,j}$.
Comentariile sunt precedate de două bare oblice // înainte și continuă până la sfârșitul liniei
Pentru operații aritmetice se utilizează simboluri standard ale operatorilor aritmetici: + (adunare), – (scădere), * (înmulțire), / (împărțire), ^ (ridicare la putere), √ (rădăcină pătrată), pentru partea întreagă a unui număr real se utilizează încadrarea expresiei între paranteze drepte, iar pentru restul împărțirii a două numere întregi se utilizează operatorul %.

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

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

pentru C++:

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

pentru Python și C++:

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

Instrumente online pentru implementarea soluțiilor

- Online GDB (https://www.onlinegdb.com)
- Programiz (https://www.programiz.com/)
- w3schools (https://www.w3schools.com/)
- Repl.it / Jupyter Notebooks (inclusiv pentru activități de programare colaborativă)
- 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 internă (sau locală) 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.

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.;
- 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
- 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/) sau Machine Learning for Kids (https://machinelearningforkids.co.uk) pentru a antrena modele simple (clasificare de imagini, recunoaștere de sunete), conectând activitățile la domenii diverse (biologie, arte, științe sociale)

GRUP DE LUCRU

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

Nume și prenume	Grad didactic/Titlu științific	Instituție de apartenență, localitate, județ
NIȚU Alina	profesor, gradul didactic I	Liceul Teoretic „Ovidius”, Constanța, județul Constanța
NURLA Aidan	profesor, gradul didactic I	Colegiul Naţional Militar „Alexandru Ioan Cuza”, Constanţa, județul Constanța
OANCEA Romana	conferențiar universitar, doctor	Academia Forţelor Terestre „Nicolae Bălcescu”, Sibiu, județul Sibiu
PETRESCU Manuela-Andreea	lector universitar, doctor	Universitatea Babeș-Bolyai,Cluj-Napoca, județul Cluj
PINTEA Adrian-Doru	profesor, gradul didactic I	Colegiul Național „Andrei Mureșanu”, Dej, județul Cluj
PINTEA Rodica	profesor, gradul didactic I	Liceul Teoretic „Radu Vlădescu”, Pătârlagele, județul Buzău
POP Florin	profesor universitar, doctor	Universitatea Națională de Științe și Tehnologie Politehnica București, Facultatea de Automatică și Calculatoare
POSTOLACHE Florin	lector universitar, doctor	Academia Navală „Mircea cel Bătrân” Constanța, Facultatea de Inginerie Marină, județul Constanța
SĂCUIU Silviu Eugen	profesor, gradul didactic I	Colegiul Național „Mihai Viteazul”, București
SPĂTARU Adrian	lector universitar, doctor	Universitatea de Vest, Timișoara, Facultatea de Matematică și Informatică, județul Timiș
STANCIU Diana	profesor, gradul didactic I	Colegiul Național 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 de Curriculum și Evaluare
ȚOCA Livia Demetra	consilier	Centrul Național de Curriculum și Evaluare
ȚĂPUS Nicolae	profesor universitar, doctor	Academia Română
BORIGA Radu Eugen	conferențiar universitar, doctor	Universitatea București, Facultatea de Matematică și Informatică