SCRISOARE METODICĂ – START ÎN CLASA a IX-a

Repere pentru înțelegerea schimbării curriculare și organizarea primelor 7 săptămâni de clasa a IX-a — disciplina Informatică, curriculum de specialitate, filiera teoretică, profilul real, specializarea Matematică-Informatică , clase cu predarea disciplinei informatică în regim intensiv

Informatică, clasa a IX-a, CS, Matematică-Informatică (4 ore/săptămână: 2 teoretică + 2 practică)

Proiect „REGLEMENTĂRI NOI PENTRU UN CURRICULUM RELEVANT ȘI EDUCAȚIE DESCHISĂ”
Activitatea A3 – Resurse educaționale de sprijin pentru aplicarea noului curriculum pentru liceu
A3.1 – Elaborarea și publicarea de ghiduri metodologice pentru disciplinele de trunchi comun

Varianta de față este elaborată pentru disciplina **Informatică**, clasa a IX-a, curriculum de specialitate, filiera teoretică, profilul real, specializarea **matematică-informatică – clase cu predarea Informaticii în regim intensiv**, aprobată prin Ordinul nr. 4.370/2026, cu un buget de timp de **4 ore/săptămână**: **2 ore de studiu teoretic și 2 ore de activități practice**. Activitățile practice se desfășoară obligatoriu în laboratorul de informatică, cu organizarea colectivului de elevi în grupe, în condițiile prevăzute de programa școlară.

Documentul are ca reper curricular **programa școlară pentru disciplina Informatică, clasa a IX-a, clase cu predarea disciplinei informatică în regim intensiv**. Pentru identificarea și valorificarea achizițiilor anterioare ale elevilor sunt avute în vedere programa de **Informatică și TIC pentru clasele a V-a–a VIII-a** (OMEN nr. 3.393/28.02.2017) și **standardele naționale de evaluare** aprobate prin OMEC nr. 4.615/2026 (M. Of., Partea I, nr. 674 bis/14 august 2026), anexa nr. 15. Reperele conceptuale și metodologice sunt fundamentate pe **Cadrul de referință al Curriculumului național** (OMEC nr. 4.137/2026).

În proiectarea demersului didactic pentru regimul intensiv, **cele 4 ore săptămânale nu sunt tratate ca o simplă extindere cantitativă a parcursului de la matematică-informatică**, ci ca o oportunitate de aprofundare a competențelor, creștere a ponderii activităților practice, investigare și comparare a soluțiilor, dezvoltare de proiecte și realizare a unor produse software cu un grad sporit de complexitate.

1. Rolul scrisorii metodice

Această scrisoare metodică oferă profesorului de **Informatică** repere și instrumente pentru debutul implementării noului curriculum liceal la clasa a IX-a. Demersul propus urmează o succesiune intenționată: mai întâi sunt clarificate **elementele de continuitate, dezvoltare și noutate în studiul disciplinei Informatică**; apoi sunt identificate achizițiile-ancoră formate în gimnaziu și nivelul real al elevilor, prin raportare selectivă la competențele și standardele relevante pentru noile învățări; pe această bază sunt proiectate și organizate primele experiențe de predare–învățare–evaluare.
Documentul nu constituie o planificare obligatorie și nu înlocuiește programa sau manualul școlar. Acesta are rolul unui **ghid metodologic** care sprijină profesorul să înțeleagă sensul schimbării curriculare, să valorifice continuitatea dintre gimnaziu și liceu și să adapteze demersul didactic la nivelul real și la nevoile de învățare ale elevilor. În acest scop, sunt oferite exemple de situații de învățare, instrumente de diagnoză și repere pentru fundamentarea deciziilor privind proiectarea, predarea, evaluarea, diferențierea și ajustarea parcursului de învățare.

Schimbarea curriculară Programa de Informatică pentru clasa a IX-a, clase cu predarea disciplinei informatică în regim intensiv , valorifică achizițiile de algoritmică și programare formate în gimnaziu și le dezvoltă către un nivel sporit de complexitate, autonomie și aprofundare. Elevul evoluează de la elaborarea și implementarea unor algoritmi elementari către modelarea și organizarea datelor, analiza și compararea strategiilor de rezolvare, proiectarea modulară, implementarea, testarea și depanarea soluțiilor, precum și către evaluarea corectitudinii și eficienței acestora. Bugetul de timp specific regimului intensiv permite o pondere mai mare a activităților practice, a investigării unor soluții alternative și a dezvoltării unor aplicații cu un grad de complexitate mai ridicat.
O dimensiune importantă a tranziției către clasa a IX-a o reprezintă armonizarea limbajului de programare. Competențele algoritmice dobândite anterior constituie punctul de plecare, iar trecerea la limbajul prevăzut de programa pentru regimul intensiv se realizează prin transfer, nu prin reluarea sistematică a algoritmicii elementare. Profesorul valorifică ceea ce elevii știu deja despre date, variabile, expresii, structuri de control, algoritmi, testare și depanare și introduce progresiv sintaxa și particularitățile noului mediu de implementare. Consolidarea unor achiziții de bază este justificată atunci când dovezile obținute prin diagnoza inițială indică această nevoie.
În regim intensiv, procesul de rezolvare a problemelor este abordat în integralitatea sa și aprofundat progresiv: problemă → modelarea și organizarea datelor → identificarea și compararea strategiilor → proiectarea soluției → implementarea → testarea și depanarea → analiza corectitudinii și eficienței → optimizarea și îmbunătățirea soluției.
Este esențială, în acest context, distincția dintre competența algoritmică și limbajul de programare. Un elev care a dobândit în gimnaziu competențe de programare într-un anumit limbaj nu pornește de la zero în clasa a IX-a. El transferă achizițiile privind reprezentarea datelor, expresiile, structurile de control, proiectarea algoritmilor, testarea și depanarea și le utilizează în contexte mai complexe. În regimul intensiv, acest transfer este continuat prin sarcini care solicită analiză, modularizare, compararea soluțiilor, justificarea alegerilor și evaluarea eficienței.
Prin urmare, specificul regimului intensiv nu constă în simpla adăugare a unor conținuturi sau a unui număr mai mare de probleme, ci în creșterea treptată a complexității sarcinilor, a autonomiei elevului și a profunzimii demersului algoritmic. Timpul suplimentar permite trecerea de la aplicarea unei soluții cunoscute la investigarea, compararea, argumentarea, optimizarea și elaborarea unor soluții proprii, cu accent pe calitatea procesului de dezvoltare a produsului software.

CE SUPLIMENTEAZĂ	CUM
Înțelegerea schimbărilor curriculare și a continuității între gimnaziu și liceu	**Harta disciplinară „Ce se păstrează – Ce se dezvoltă/se schimbă – Ce înseamnă la clasă”**, care pune în relație achizițiile de algoritmică și programare formate în gimnaziu cu noile cerințe ale programei de Informatică pentru clasa a IX-a: modelarea și organizarea datelor, alegerea și compararea strategiilor de rezolvare, utilizarea algoritmilor specializați, modularizarea soluțiilor, testarea și evaluarea corectitudinii și eficienței, precum și proiectarea unor soluții informatice coerente.

CE SUPLIMENTEAZĂ	CUM
Resursele oferite de manualul școlar	**Un pachet de resurse utilizabile imediat**, complementar manualului: enunțuri de probleme, seturi de date, exemple și contraexemple, fragmente de algoritmi și programe, fișe de lucru, sarcini de investigare, testare și depanare, instrumente de evaluare și resurse digitale verificate.
Reperele metodologice necesare în momentele-cheie ale deciziei didactice	**Orientări metodologice integrate în demersul didactic specific disciplinei Informatică** asociate etapelor procesului de rezolvare: analiza problemei, modelarea și organizarea datelor, proiectarea algoritmului, implementarea, testarea, depanarea, justificarea și evaluarea soluției. Reperele răspund unor întrebări operaționale: • **ce fac?** – selectez problema/sarcina și organizez procesul de rezolvare; • **de ce fac?** – raportez activitatea la competența vizată și la tipul de raționament algoritmic urmărit; • **la ce mă uit?** – observ modul în care elevul analizează problema, modelează datele, construiește algoritmul, implementează și testează soluția și își justifică alegerile; • **ce ajustez?** – adaptez gradul de complexitate al sarcinii, modalitatea de reprezentare a datelor, nivelul de sprijin oferit, exemplele și contraexemplele, seturile de date utilizate pentru testare și cerințele privind îmbunătățirea soluției.
Tranziția de la practicile centrate predominant pe parcurgerea conținuturilor la practici centrate pe dezvoltarea competențelor.	**Exemple comentate de activități didactice**, care deplasează accentul de la reproducerea sintaxei și aplicarea mecanică a unor algoritmi către situații de învățare în care elevul analizează problema, identifică și organizează datele, compară strategii de rezolvare, proiectează și implementează soluții, testează și depanează programele și argumentează alegerile realizate.
Cunoașterea nivelului real al achizițiilor elevilor la intrarea în clasa a IX-a.	**Instrumente de diagnoză și exemple de diferențiere**, construite în jurul achizițiilor-ancoră din gimnaziu: analiza problemei, identificarea datelor de intrare și de ieșire, utilizarea variabilelor și expresiilor, aplicarea structurilor de control, urmărirea execuției unui algoritm, implementarea, testarea și depanarea. Dovezile obținute prin diagnoză fundamentează adaptarea sarcinilor, a nivelului de sprijin și a gradului de complexitate la nevoile reale de învățare ale elevilor.

Harta disciplinară pentru disciplina Informatică, clasa a IX-a

Domeniu	Ce se păstrează și se valorifică din gimnaziu	Ce se dezvoltă/se aprofundează în clasa a IX-a	Ce înseamnă la clasă
Analiza și rezolvarea problemelor	Identificarea datelor de intrare și de ieșire, stabilirea prelucrărilor necesare și elaborarea unui algoritm de rezolvare.	Analiza structurată și modelarea problemei, identificarea și compararea unor strategii posibile de rezolvare.	Elevul analizează problema înainte de implementare: identifică datele, condițiile sau restricțiile problemei și rezultatele așteptate, explorează variante de rezolvare și argumentează strategia aleasă.

Domeniu	Ce se păstrează și se valorifică din gimnaziu	Ce se dezvoltă/se aprofundează în clasa a IX-a	Ce înseamnă la clasă
Modelarea și organizarea datelor	Utilizarea variabilelor, constantelor, operatorilor și expresiilor pentru reprezentarea și prelucrarea datelor.	Alegerea și utilizarea unor modalități adecvate de reprezentare, organizare și prelucrare a datelor, în funcție de problema care trebuie rezolvată.	Elevul stabilește modul de reprezentare și organizare a datelor și justifică alegerea făcută în raport cu cerințele problemei.
Strategii de rezolvare a problemelor	Construirea algoritmilor utilizând algoritmi de bază pentru prelucrarea seriilor de valori. Aplicarea unor parcurgeri și a unor prelucrări algoritmice elementare în contexte familiare.	Studierea și utilizarea strategiilor generale de proiectare modulară a soluțiilor și a algoritmilor specializați pe clase de probleme pentru prelucrarea numerelor și a colecțiilor de date, precum și pentru generarea sistematică, căutarea și sortarea.	Elevul identifică tipul problemei, selectează, adaptează și aplică algoritmul adecvat și transferă strategia de rezolvare în contexte noi.
Memorarea datelor și organizarea codului în limbaj de programare	Utilizarea datelor și realizarea unor operații elementare de citire, prelucrare și stocare în variabile simple, într-un limbaj ales.	Utilizarea, listelor în limbajul Python și a tablourilor unidimensionale în limbajul C++ și a fișierelor text pentru organizarea, stocarea și prelucrarea datelor, precum și comunicarea cu programul prin interfețe grafice proiectate.	Elevul proiectează soluții în care datele sunt organizate, citite, prelucrate și stocate în concordanță cu cerințele problemei.
Strategii de rezolvare a problemelor. Principii de elaborare a unui program	Caracteristici ale principalelor etape ale elaborării unui program Descompunerea unei probleme în pași și secvențe de operații necesare pentru rezolvare. Caracteristici de bază ale unor moduri de reprezentare a algoritmilor (blocuri grafice, limbaj de programare) Verificarea rezultatelor obținute pentru diferite date de intrare și identificarea situațiilor în care soluția trebuie corectată. Executarea programelor, urmărirea valorilor variabilelor, identificarea și corectarea erorilor.	Proiectarea și realizarea unor soluții informatice coerente, modulare și adecvate cerințelor problemei. Descompunerea problemei în subprobleme și utilizarea subprogramelor pentru proiectarea unor soluții modulare și reutilizabile. Caracteristici de bază, avantaje și limite ale principalelor moduri de reprezentare a algoritmilor Criterii de elaborare a testelor Analiza corectitudinii și eficienței soluțiilor și compararea unor strategii sau algoritmi pentru aceeași problemă. Proiectarea sistematică a testelor și utilizarea acestora pentru verificarea și îmbunătățirea soluției.	Elevul parcurge și documentează un proces complet de rezolvare: **analiză → modelare → proiectare → implementare → testare → evaluare → îmbunătățire**. Elevul structurează soluția în componente cu roluri bine definite, stabilește relațiile dintre acestea și le integrează într-o soluție coerentă. Elevul argumentează corectitudinea soluției pentru cazurile relevante, compară alternativele și identifică posibilități de îmbunătățire. Elevul construiește seturi de teste, inclusiv pentru cazuri particulare și cazuri-limită, identifică sursa erorilor și revizuiește algoritmul sau implementarea.

La începutul clasei a IX-a, profesorul care predă disciplina Informatică își fundamentează demersul didactic pe trei repere: cerințele noului curriculum, achizițiile algoritmice relevante formate în gimnaziu și nivelul real de competență evidențiat de elevi în situații concrete de rezolvare a problemelor. Astfel, punctul de plecare nu este o apreciere generală privind ceea ce elevii „știu" sau „nu știu" despre programare, ci dovezile privind modul în care aceștia analizează o problemă, construiesc o strategie de rezolvare și o transpun într-o soluție algoritmică.

Primele șapte săptămâni asigură continuitatea dintre achizițiile anterioare și noile învățări, susținând progresia de la valorificarea strategiilor algoritmice deja formate către modelarea și organizarea datelor, proiectarea și implementarea soluțiilor, precum și testarea, evaluarea și îmbunătățirea acestora.

Debutul clasei a IX-a nu presupune reluarea sistematică a noțiunilor de algoritmică și programare studiate în gimnaziu, nici parcurgerea automată, în ordine, a conținuturilor din programă. Punctul de plecare îl reprezintă corelarea cerințelor noului curriculum cu achizițiile anterioare și cu nivelul real de competență al elevilor.

Înainte de proiectarea demersului didactic, profesorul își fundamentează deciziile pe răspunsurile la trei întrebări esențiale:

- Ce se păstrează, ce se dezvoltă și ce aduce nou curriculumul de Informatică pentru clasa a IX-a?
- Care sunt achizițiile-ancoră formate în gimnaziu pe care se pot construi noile învățări?
- Ce poate face efectiv fiecare elev atunci când analizează o problemă, elaborează o strategie de rezolvare, proiectează și implementează un algoritm, îl testează și își argumentează soluția?

Primele șapte săptămâni valorifică răspunsurile la aceste întrebări pentru a asigura continuitatea învățării, consolidarea selectivă a achizițiilor necesare, dezvoltarea progresivă a noilor competențe, diferențierea activităților, monitorizarea progresului și ajustarea demersului didactic pe baza dovezilor de învățare.

2. Înainte de proiectare: ce se schimbă la disciplina Informatică, clasa a IX-a, curriculum de specialitate

Pentru a proiecta coerent începutul clasei a IX-a, profesorul are nevoie de o lectură integrată a noii programe, dincolo de inventarierea competențelor și a conținuturilor. Un prim rol al anexei este acela de a clarifica sensul schimbării curriculare: ce se păstrează și se valorifică, ce se dezvoltă sau se reformulează, ce elemente noi sunt introduse, ce își modifică ponderea și, mai ales, cum se reflectă aceste schimbări în proiectarea, predarea, învățarea și evaluarea la disciplina Informatică.

2.1. Harta schimbării curriculare — Informatică, clasa a IX-a

COORDONATA SCHIMBĂRII	CE PĂSTRĂM / CONTINUĂM	CE ESTE NOU / DIFERIT ÎN CLASA a IX-a	CE ÎNSEAMNĂ PENTRU PROFESOR	SUPORTUL OPERAȚIONAL (ÎN ANEXĂ)
Finalitățile și identitatea disciplinei	Informatica din liceu valorifică achizițiile formate în cadrul disciplinei *Informatică și TIC* din gimnaziu, în special cele	Disciplina capătă un rol accentuat de **specializare**, cu 4 ore/săptămână la specializarea Matematică-Informatică clase cu predarea informaticii în regim intensiv – 2 ore pentru studiul	Activitățile teoretice și practice se proiectează integrat, ca etape complementare ale aceluiași proces de învățare: conceptele	Fișa 4.1 „Ce aduc cu mine / Ce construiesc de aici" și calendarul din secțiunea 3

	referitoare la rezolvarea algoritmică a problemelor și la programare. Caracterul aplicativ al disciplinei și dezvoltarea gândirii algoritmice rămân repere importante.	teoretic și 2 ore pentru activități practice. Se consolidează dezvoltarea gândirii computaționale, algoritmice, analitice și creative, prin abordarea unor probleme cu un grad progresiv de complexitate și prin utilizarea modelelor și instrumentelor specifice informaticii.	și modelele sunt explorate, aplicate, testate și validate prin activități practice. Programa, competențele și progresia învățării constituie reperele principale ale proiectării didactice.	
Competențe și progresia învățării	Sunt valorificate competențele dezvoltate în gimnaziu: 1. Utilizarea responsabilă și eficientă a tehnologiei informației și comunicațiilor 2. Rezolvarea unor probleme elementare prin metode intuitive de prelucrare a informației 3. Elaborarea creativă de mini proiecte care vizează aspecte sociale, culturale și personale, respectând creditarea informației și drepturile de autor	Programa structurează progresia prin șase competențe generale: **CG1** – Identifică principalele caracteristici ale modelelor conceptuale și operaționale ale dezvoltării produselor software pentru înțelegerea fundamentelor programării **CG2** – Explică principiile care stau la baza modelelor conceptuale și operaționale ale dezvoltării produselor software pentru a fundamenta în mod logic proiectarea și implementarea soluțiilor informatice **CG3** – Utilizează modelele 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	Progresia se proiectează pe verticala CG: aceeași temă se poate cere la nivel de identificare (CS 1.x), de explicare (CS 2.x), de utilizare (CS 3.x), de analizare (CS 4.x), de evaluare (CS 5.x) sau de elaborare (CS 6.x). Profesorul urmărește dezvoltarea competenței, nu parcurgerea mecanică a conținuturilor. Aceeași problematică poate fi abordată la niveluri cognitive diferite: elevul identifică și explică, apoi utilizează, analizează, evaluează și elaborează soluții. Activitățile inițiale trebuie să valorifice achizițiile anterioare și să creeze treptat contexte pentru analiză, testare și argumentare.	Anexa privind corelarea competențelor, a activităților de învățare și a dovezilor de progres.

Conținuturi și limbaj de programare	Se valorifică noțiunile de algoritm, date de intrare și de ieșire, variabile, expresii, structuri secvențiale, alternative și repetitive, precum și strategiile algoritmice elementare formate anterior.	În programa pentru clase cu predarea disciplinei informatică în regim intensiv, limbajul de programare Python este studiat ca bază, în timp ce **C++ se menține ca al doilea limbaj.** Se dezvoltă achiziții privind organizarea și prelucrarea datelor, fișierele text, modele liniare de organizare a datelor, subprogramele-modularizarea, reutilizarea codului și evaluarea eficienței, în acord cu programa pentru clasa a IX-a. Astfel în clasa a IX-a sunt nou introduse conținuturi precum: - modele conceptuale explicite pentru organizarea datelor ca liste și tablouri unidimensionale - elemente de limbaj Python și C++ pentru memorarea și prelucrarea datelor: subprograme, fișiere text, introducere în POO, clasa list, biblioteca Tkinter pentru interfețe grafice, - strategii de rezolvare a problemelor: eficiența algoritmilor (notația O), metode specializate pe clase de probleme (de exemplu, generare sistematică de valori, sortare prin selecția minimului, liste de frecvențe, sortare prin metoda bulelor; prelucrări ale datelor numerice).	Conținuturile sunt utilizate ca **resurse pentru dezvoltarea competențelor**. Profesorul valorifică gândirea algoritmică formată anterior și sprijină transferul către limbajele Python și/sau C++ Se realizează armonizarea achizițiilor și remedierea punctuală a dificultăților, fără reluarea sistematică a întregii algoritmici de bază.	Activități de transfer către limbajele Python și C++ prin exemple comparative de reprezentare, implementare, testare și depanare.
Activități și experiențe de învățare	Se păstrează rezolvarea de probleme, exercițiul algoritmic, lucrul la calculator, testarea soluțiilor și realizarea unor aplicații ca modalități de învățare.	Este accentuat procesul complet de elaborare a unei soluții informatice: **analiză → modelare → proiectare → implementare → testare și depanare → evaluare → îmbunătățire.** Exemplele de activități de învățare (EAI) din programă au un caracter orientativ și pot fi adaptate sau înlocuite în funcție de contextul clasei.	Elevul este implicat în întregul proces de rezolvare: **analizează, proiectează, implementează, testează, corectează și argumentează**. Profesorul creează sarcini care produc dovezi observabile ale învățării: algoritmi, programe,	Secvențe de învățare; fișe de laborator; sarcini diferențiate și exemple de proiecte aplicative.

		Se recomandă explicit: învățare prin descoperire, proiect, portofoliu, mentorat între elevi, învățare contextualizată; Pentru facilitarea înțelegerii execuției algoritmilor implementați în Python sau C++, pot fi utilizate instrumente de vizualizare și simulare, precum Python Tutor, VisuAlgo sau Algorithm Visualizer, care permit urmărirea pas cu pas a execuției, a modificării valorilor variabilelor și, după caz, a funcționării structurilor de date și a algoritmilor.	seturi de teste, explicații ale soluției, comentarii asupra erorilor și propuneri de îmbunătățire.	
Evaluare și feedback	Se valorifică metodele tradiționale și moderne de evaluare prin soluții realizate, programe și portofolii.	**Standardele naționale de evaluare (OMEC nr. 4.615/2026)** oferă repere pentru descrierea progresului elevilor pe trei niveluri de performanță – **de bază**, **consolidat** și **avansat** –, diferențiate în funcție de complexitatea sarcinii, de gradul de autonomie și de noutatea contextului. Pentru învățământul gimnazial, aceste standarde sunt deja aprobate, prin anexa nr. 15, constituind repere utile pentru identificarea achizițiilor cu care elevii intră în clasa a IX-a. Programa pentru clasa a IX-a acordă un rol important evaluării formative, integrate pe parcursul învățării. Aceasta urmărește atât rezultatul obținut, cât și procesul prin care elevul ajunge la soluție: modul în care analizează problema, formulează și aplică strategii de rezolvare, proiectează și testează algoritmi, identifică erori și își revizuiește soluția. Evaluarea sumativă, realizată în raport cu competențele vizate și cu dovezile de învățare relevante, completează evaluarea formativă și oferă repere pentru aprecierea progresului și pentru proiectarea etapelor următoare ale învățării.	Evaluarea vizează atât procesul de rezolvare, cât și produsul obținut: analiza problemei, strategia aleasă, proiectarea soluției, elaborarea și utilizarea seturilor de teste, urmărirea evoluției valorilor variabilelor la executarea algoritmului, identificarea și corectarea erorilor, precum și justificarea soluției finale. Criteriile de reușită sunt comunicate odată cu sarcina și formulate astfel încât să ofere elevului repere clare pentru realizarea activității, pentru autoevaluare și pentru îmbunătățirea propriei soluții. Proiectarea evaluării poate fi ghidată de întrebările-cheie formulate în	Criteriile de reușită din secțiunea 6 și instrumentul 8.2

			sugestiile metodologice ale programei: „**De ce evaluez? Ce evaluez? Cum evaluez? Cât de adecvat măsoară instrumentul ceea ce urmăresc? Ce feedback ofer? Ce decizii didactice iau pe baza rezultatelor?**	
Flexibilitate și rezerva de 25%	Adaptarea ritmului, a sarcinilor și a sprijinului oferit la nivelul real de pregătire, în funcție de progresul elevilor.	Programa este proiectată cu o **rezervă de 25% din timpul alocat disciplinei**, aflată la dispoziția profesorului pentru **remediere, consolidare, aprofundare sau extindere**, în funcție de nevoile de învățare identificate. Această rezervă oferă spațiu curricular pentru adaptarea parcursului la particularitățile clasei și la progresul elevilor.	Utilizarea rezervei de 25% se stabilește pe baza dovezilor de învățare obținute prin diagnoză, observare, evaluare formativă și analiza produselor realizate de elevi. Profesorul decide, în funcție de nevoile identificate, dacă timpul disponibil este utilizat pentru remedierea unor achiziții esențiale, consolidarea celor insuficient stabilizate, aprofundarea unor competențe sau extinderea învățării prin sarcini cu un grad sporit de complexitate. În primele șapte săptămâni, decizia privind utilizarea rezervei se fundamentează pe datele colectate pe parcursul activităților și al evaluărilor, fără a se stabili anticipat o destinație unică.	**Scenarii orientative de utilizare a rezervei de 25%** – remediere, consolidare, aprofundare și extindere – corelate cu dovezile de învățare și cu nevoile identificate; secțiunea 3 și reperele pentru diferențiere din secțiunea 6.

Conexiuni interdisciplinare, competențe digitale, inteligență artificială și contexte actuale	Se valorifică legăturile cu alte discipline, de exemplu, cu **matematica**, pentru noțiuni specifice (de exemplu, pentru divizibilitate), raționament logic, formalizare, modelare, reprezentarea datelor și elaborarea unor strategii de rezolvare a problemelor.	Se extinde valorificarea informaticii în **contexte interdisciplinare și autentice**, prin conexiuni cu domenii STEM, economie, științe sociale, arte digitale și cu teme transversale relevante. Contexte tehnologice actuale, inclusiv cele asociate **inteligenței artificiale, învățării automate și utilizării datelor**, pot oferi situații relevante pentru aplicarea gândirii computaționale și algoritmice.	Conexiunile interdisciplinare sunt integrate prin contextul și datele problemei, fără a deveni conținuturi suplimentare desprinse de competențele disciplinei. Profesorul selectează situații autentice în care elevii aplică achizițiile de la disciplina informatică pentru a analiza și organiza date, a identifica modele, a proiecta algoritmi, a implementa și testa soluții și a interpreta rezultatele. Contextele sunt alese în funcție de competențele vizate, de relevanța lor pentru elevi și de oportunitățile reale de integrare curriculară.	**Context autentic de învățare:** prelucrarea și analiza unui set de date reale – de exemplu, temperaturi, consum, măsurători de mediu, rezultate sportive sau indicatori economici – prin parcurgerea etapelor **analiză → modelare → proiectare → implementare → testare → interpretarea rezultatelor**

2.2. „Ce trebuie să văd altfel de luni dimineață, 07.09?” – mesaje-cheie pentru profesor

MESAJ-CHEIE	EXPLICAȚIE ÎN LIMBAJ PROFESIONAL SIMPLU	MIC EXEMPLU DIN DISCIPLINA INFORMATICĂ
M1. Se recomandă ca la începutul clasei a IX-a să se realizeze o testare inițială, pentru o diagnoză a nivelului de valorificare a achizițiilor anterioare în contexte de rezolvare a problemelor.	Debutul clasei a IX-a valorifică achizițiile din gimnaziu în cadrul noilor situații de învățare. Se pot propune de la început sarcini care permit observarea modului în care elevii analizează datele, construiesc algoritmi, implementează și testează soluții. Dificultățile identificate sunt abordate punctual, prin consolidare sau prin remediere.	În săptămâna 1, elevii primesc o fișă pe baza căreia se face o diagnoză a achizițiilor din gimnaziu.

M2. **Facilitez transferul către Python și C++,** valorificând gândirea algoritmică deja formată, și predau tema Strategii de rezolvare a problemelor. Principii de elaborare a unui program	Utilizarea limbajelor **C++ și Python** nu presupune reluarea algoritmicii de bază și nici predarea separată a acestora ca succesiuni de instrucțiuni. Profesorul pornește de la **gândirea algoritmică și experiențele de programare deja formate**, evidențiază corespondențele dintre modalitățile de implementare a acelorași structuri algoritmice și sprijină transferul achizițiilor de la un limbaj la altul. Nivelul de ghidaj, complexitatea sarcinilor și ponderea utilizării fiecărui limbaj se adaptează competențelor vizate și experienței reale a elevilor.	Se recomandă ca, în primele săptămâni, soluțiile să fie implementate atât în Python/C++, cât și în limbajul utilizat la gimnaziu, în pseudocod și scheme logice, identificând elementele comune și diferențele de implementare.
M3. **În activitățile practice colectez dovezi ale procesului de învățare, nu doar produsul final.**	Evaluarea formativă urmărește modul în care elevul analizează problema, proiectează soluția, o implementează, o testează, identifică erorile și își revizuiește demersul. Orice activitate practică poate genera **dovezi intermediare și finale ale învățării**: reprezentarea algoritmului, fragmente de cod, programe funcționale, seturi de teste, explicații ale alegerilor făcute sau jurnale de depanare.	Într-o activitate practică, elevul construiește și justifică un **set de teste** care include situații uzuale și cazuri-limită. Setul de teste devine o dovadă a înțelegerii problemei și este utilizat ulterior pentru verificarea și îmbunătățirea programului.
M4. **Programa este reperul curricular;** flexibilitatea se construiește pe baza dovezilor de învățare.	Proiectarea pornește de la competențele specifice, conținuturile și sugestiile metodologice ale programei, iar resursele didactice sunt selectate sau elaborate în funcție de acestea și de particularitățile clasei. Rezerva de 25% oferă profesorului spațiu pentru **remediere, consolidare, aprofundare sau extindere**, iar utilizarea ei se decide pe baza nevoilor evidențiate pe parcursul învățării, nu prin stabilirea anticipată a unei destinații fixe.	După o secvență de învățare, profesorul analizează rezultatele sarcinilor, produsele elevilor, erorile recurente și autoevaluările. Pe baza acestor dovezi, proiectează activități diferențiate: - **remediere** pentru achizițiile insuficient formate, - **consolidare** pentru stabilizarea acestora sau - **aprofundare/extindere** prin abordarea unor probleme mai complexe și explorarea unor soluții alternative.

2.3. Deciziile profesorului, dincolo de manual

Demersul didactic pornește de la programă și de la nivelul real al elevilor. Manualul și celelalte resurse susțin acest proces, dar deciziile privind proiectarea, predarea, evaluarea și ajustarea învățării revin profesorului.

Moment	CE FACE PROFESORUL
1. Înțeleg schimbarea curriculară	Analizează programa și harta schimbării pentru a identifica ce se păstrează, ce se dezvoltă și ce este nou în clasa a IX-a, precum și implicațiile acestor schimbări asupra predării, învățării și evaluării.
2. Identific achizițiile-ancoră	Selectează achizițiile relevante din gimnaziu pe care se construiesc noile învățări, fără a transforma începutul anului într-o recapitulare extinsă.
3. Colectez dovezi despre nivelul inițial	Utilizează o diagnoză scurtă și variată, bazată pe sarcini relevante, pentru a observa ce pot face efectiv elevii și unde este necesar sprijin suplimentar.
4. Proiectez unitatea de învățare	Corelează competențele specifice și conținuturile → activitățile de învățare → activitățile și instrumentele de evaluare → dovezile de învățare, urmărind progresia competențelor.
5. Selectez și pregătesc resursele	Alege resursele strict necesare pentru realizarea sarcinilor: probleme, seturi de date, exemple și contraexemple, fragmente de cod, fișe de lucru, medii de programare și instrumente de evaluare.
6. Facilitez învățarea și colectez dovezi	Urmărește procesul de rezolvare, oferă feedback pe parcurs și colectează dovezi privind analiza problemei, proiectarea, implementarea, testarea, depanarea și argumentarea soluției.
7. Ajustez demersul didactic	Utilizează dovezile colectate pentru a adapta sarcinile, nivelul de sprijin, ritmul și gradul de complexitate, precum și pentru a decide activitățile de remediere, consolidare, aprofundare sau extindere, inclusiv prin valorificarea rezervei de 25%.

3. Arhitectura primelor 7 săptămâni

Primele șapte săptămâni propun un parcurs care integrează valorificarea achizițiilor din gimnaziu, diagnoza, familiarizarea cu limbajele Python și C++ și dezvoltarea progresivă a întregului proces de rezolvare a problemelor. Succesiunea este orientativă și poate fi ajustată de profesor în funcție de nivelul inițial al elevilor și de dovezile de învățare colectate pe parcurs.

Ritmul și gradul de aprofundare se adaptează timpului alocat disciplinei: **4 ore/săptămână la specializarea Matematică-informatică cu clase de informatică intensiv,** din care 2 ore sunt alocate activităților teoretice și 2 ore activităților practice în laborator.

SĂPTĂMÂNA	FOCUS	ÎNTREBAREA PROFESORULUI	PRODUS / EVIDENȚĂ PRINCIPALĂ
1	**L1. Diagnoza inițială și principii de elaborare a unui program** **Diagnoza inițială poate fi integrată în activități didactice**, prin sarcini scurte care reactivează achizițiile-ancoră din gimnaziu.(sarcinile D1–D3). Pe durata primei săptămâni începe predarea temei „Strategii de rezolvare a problemelor". Principiile de elaborare a unui program. Elevii primesc, pe parcurs, sarcini care pun în evidență etapele: analiză → proiectare → implementare → testare. Rezolvările vor fi reprezentate în paralel în schemă logică, pseudocod, limbajul utilizat în gimnaziu, limbajele Python și C++, asigurându-se astfel armonizarea acestor moduri de reprezentare. Problemele sunt alese astfel încât să fie reactivate achizițiile din gimnaziu privind instrucțiunile, algoritmii de bază, datele simple. Analiza problemei, implementarea, testarea și depanarea se vor relua repetitiv pe parcursul studiului disciplinei. La finalul temei, elevii realizează un tabel comparativ cu trei forme de reprezentare — blocuri grafice | pseudocod, limbajul utilizat la gimnaziu, și| Python — pentru structurile alternative și repetitive, pornind de la algoritmi cunoscuți din gimnaziu și identificând corespondențele dintre acestea. Exemple de fișe de lucru care pot fi utilizate: E1, E2.	Cum abordează elevii o problemă de informatică și ce achiziții anterioare pot valorifica? În ce măsură identifică datele, formulează o strategie, utilizează structuri algoritmice cunoscute și verifică rezultatul? Recunosc elevii aceeași logică algoritmică în reprezentări diferite? Ce dificultăți apar în armonizarea cu limbajele Python și C++ și ce sprijin este necesar?	1.Fișa de analiză a problemei: date de intrare/ieșire, condiții și rezultat așteptat 2.Programul realizat 3.Răspunsurile la sarcinile de diagnoză. 4.Tabelul comparativ al celor patru reprezentări: cu blocuri grafice → pseudocod → un limbaj de programare utilizat la gimnaziu → Python → C++

SĂPTĂMÂNA	FOCUS	ÎNTREBAREA PROFESORULUI	PRODUS / EVIDENȚĂ PRINCIPALĂ
2, 3, 4	U2. L1 Strategii de rezolvare a problemelor.Prelucrări ale datelor numerice - operații specifice cu cifrele unui număr (acces la o cifră a unui număr, adăugare a unei cifre la stânga unui număr, adăugare a unei cifre la dreapta unui număr) - repere pentru prelucrarea numerelor (parcurgerea cifrelor unui număr) și aplicarea algoritmilor de bază; - repere pentru transformarea unui număr dintr-o bază de numerație în altă bază de numerație; 1. Exemplu de activitate **practică (fișa E3)**: proiectarea, implementarea și testarea unor algoritmi pentru prelucrarea cifrelor unui număr.	Transferă elevul către procesul complet de rezolvare – analiză → proiectare → implementare → testare → evaluare – într-un context nou, cu un grad mai redus de ghidare?	Soluții algoritmice implementate și testate Seturi de teste proprii 1. Fișă de analiză prin care elevul explică strategia utilizată și verifică soluția.
4, 5, 6	U2.L2 Strategii de rezolvare a problemelor. Prelucrări ale datelor numerice -determinarea unui divizor/multiplu al unui număr; - repere pentru prelucrarea numerelor (parcurgerea 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); Exemple de activități practice (fișe E4 și E5): proiectarea, implementarea și testarea unor algoritmi pentru prelucrarea divizorilor unui număr.	Transferă elevul către procesul complet de rezolvare – analiză → proiectare → implementare → testare → evaluare – într-un context nou, cu un grad mai redus de ghidare?	Soluții algoritmice implementate și testate Seturi de teste proprii Fișă de analiză prin care elevul explică strategia utilizată și verifică soluția.
7	Consolidare, remediere, extindere Exemple de activități practice (fișe E6, E7, E8, E9): proiectarea, implementarea și testarea unor algoritmi pentru prelucrarea divizorilor unui număr.	Asigură consolidare, remediere și extindere?	Soluții algoritmice implementate și testate Seturi de teste proprii Fișă de analiză prin care elevul explică strategia utilizată și verifică soluția

Calendar orientativ pentru primele 7 săptămâni – Matematică-informatică și Științe ale naturii
Calendarul este construit diferențiat, în funcție de timpul alocat disciplinei: **4 ore/săptămână la specializarea Matematică-informatică (28 de ore în 7 săptămâni)**.
Pentru specializarea **Matematică-informatică**, parcursul debutează cu domeniul diagnoza inițială și *Strategii de rezolvare a problemelor*, prin abordarea principiilor de elaborare a unui program. Achizițiile algoritmice și de programare formate în gimnaziu nu sunt reluate într-o secvență distinctă de recapitulare, ci reactivate și valorificate în contexte de rezolvare a problemelor. Cele 4 ore săptămânale sunt proiectate integrat, astfel încât conceptualizarea, analiza și proiectarea soluției să fie continuate prin activități practice de implementare, testare și depanare în laborator.
Rezerva de 25% nu este asociată anticipat cu o săptămână fixă. Utilizarea acesteia se decide pe baza **dovezilor de învățare** colectate pe parcurs și poate susține, în funcție de nevoile identificate, activități de remediere, consolidare, aprofundare sau extindere. Calendarul are, prin urmare, un **caracter orientativ și flexibil**, putând fi ajustat în funcție de nivelul real al elevilor și de progresul observat.

4. Cultura disciplinei

4.1. Fișa „Ce aduc cu mine / Ce dezvolt în clasa a IX-a”

Fișa sprijină profesorul în identificarea **achizițiilor din gimnaziu** relevante pentru noile învățări și în proiectarea unor activități prin care acestea sunt reactivate, verificate și dezvoltate. Verificarea nu urmărește inventarierea tuturor conținuturilor studiate anterior, ci identificarea a ceea ce elevul **poate utiliza efectiv într-o situație de rezolvare a unei probleme**.

ANCORE DIN GIMNAZIU	CUM ÎMI DAU SEAMA (VERIFICAREA ACHIZIȚIILOR)	ABORDARE NOUĂ LA LICEU / CUM SE DEZVOLTĂ ÎN CLASA a IX-a	PRIMUL PAS ÎN ÎNVĂȚARE
A1. Date, variabile, date de intrare și de ieșire. Elevul a lucrat cu noțiunile de algoritm, tipuri de date, constante și variabile, precum și cu operații de citire, prelucrare și afișare a datelor.	Reper din standardele de evaluare (SNE, anexa nr. 15, clasa a V-a, CS 2.2): elevul *analizează situații algoritmice date pentru a identifica și clasifica datele utilizate de algoritmul corespunzător*. Verificare: **sarcina D1** din secțiunea 5 – pentru o problemă dată, identifică datele de intrare, rezultatele așteptate și valorile intermediare necesare pentru rezolvare.	Accentul se deplasează de la simpla identificare a datelor către modelarea și organizarea acestora în funcție de problema rezolvată. Elevul analizează ce date sunt necesare, cum trebuie reprezentate și organizate, precum și ce informații trebuie păstrate pe parcursul prelucrării. Ulterior, sunt introduse și utilizate modelele și structurile de date prevăzute de programă.	În prima problemă abordată, elevul completează **fișa de analiză: date de intrare → condiții/restricții → date de ieșire**, înainte de proiectarea și implementarea algoritmului.
A2. Structura alternativă și exprimarea deciziilor. Elevul a utilizat condiții și structuri alternative pentru construirea algoritmilor, inițial în medii grafice și ulterior în limbaj de programare.	Reper din standardele de evaluare (SNE, clasa a V-a, CS 3.2): elevul *implementează algoritmi dați în limbaj natural, care prelucrează date prin operații succesive, plasând blocuri secvențiale și alternative într-un mediu grafic interactiv*. Verificare: **sarcina D2** — completează o condiție într-un algoritm dat și anticipează rezultatul pentru mai multe seturi de date.	Condiția devine un obiect de analiză și de testare. Elevul construiește și evaluează expresii logice, analizează ramurile posibile ale execuției, identifică situații particulare și cazuri-limită și verifică, prin teste, comportamentul programului.	Elevul pornește de la o decizie cunoscută și urmărește corespondențele dintre reprezentarea algoritmică și implementarea în limbajele Python și C++, justificând condiția utilizată și verificând-o pentru date diferite.

ANCORE DIN GIMNAZIU	CUM ÎMI DAU SEAMA (VERIFICAREA ACHIZIȚIILOR)	ABORDARE NOUĂ LA LICEU / CUM SE DEZVOLTĂ ÎN CLASA a IX-a	PRIMUL PAS ÎN ÎNVĂȚARE
A3. Structura repetitivă și algoritmii de bază pe șiruri de valori(clasele a VI-a – a VIII-a: repetiții cu contor și condiționate); Elevul a utilizat structuri repetitive cu număr cunoscut sau necunoscut de pași și algoritmi elementari de numărare, însumare, calcul al produsului, determinare a minimului/maximului și verificare a unei proprietăți.	Repere din standardele de evaluare (SNE, clasa a VI-a, CS 2.3 și clasa a VIII-a, CS 2.2): elevul *elaborează algoritmi de prelucrare a informației, incluzând structuri repetitive; respectiv, elaborează programe în limbaj de programare care rezolvă probleme informatice simple de prelucrare a șirurilor de valori.* Verificare: **sarcina D3** — urmărește pas cu pas execuția unui algoritm și completează un tabel cu evoluția valorilor variabilelor, explicând rezultatul obținut.	Structurile repetitive sunt integrate în probleme cu un grad progresiv de complexitate și sunt corelate cu testarea, depanarea și analiza eficienței. Elevul compară strategii, urmărește numărul de operații și utilizează, în contexte adecvate, algoritmii specializați prevăzuți de programă.	Tabelul de evoluție a valorilor variabilelor este utilizat ca instrument de înțelegere, verificare și depanare. Ulterior, elevul compară soluțiile și argumentează diferențele dintre acestea, inclusiv prin raportare la numărul de operații efectuate.

5. Diagnoză pentru învățare, nu pentru clasificare

DOMENIU DIAGNOSTIC	ÎNTREBARE / SARCINĂ	CE URMĂRIM	DECIZIE POSIBILĂ
Repere ale învățării anterioare	**D1 – Analiza datelor unei probleme.** „O aplicație primește notele obținute de elevii unei clase la un concurs și trebuie să afișeze câți elevi au peste 8 și media acestora. Identificați datele de intrare, rezultatele așteptate și valorile intermediare necesare pentru rezolvare.”	Capacitatea elevului de a distinge între datele problemei, rezultatele urmărite și valorile necesare pe parcursul prelucrării. Se observă dacă elevul identifică elementele independent sau cu sprijin și dacă poate explica rolul acestora în construirea soluției. Nivel: identifică intrarea și ieșirea, cu sprijin = **de bază;** identifică independent și datele de manevră = **consolidat;** justifică de ce o dată trebuie memorată = **avansat**.	Dacă apar dificultăți recurente, profesorul modelează explicit analiza unei probleme și utilizează temporar o fișă de tipul **intrare → condiții/restricții → prelucrare → ieșire**. Dacă achiziția este stabilă, elevii pot trece direct la analiza în perechi sau individuală a unor probleme noi
Abilități și proceduri algoritmice	**D2 — 5 minute, pe hârtie sau la calculator.** Construirea și verificarea unei condiții. Se oferă un algoritm scurt în care lipsește condiția unei structuri alternative: „Completați condiția astfel încât algoritmul să determine maximul dintre două numere. Anticipați apoi rezultatul pentru perechile (7, 7), (–3, 2) și (5, 4), fără a executa programul.	Utilizarea operatorilor relaționali, înțelegerea logicii structurii alternative și capacitatea de a anticipa comportamentul algoritmului pentru date diferite. Cazul de egalitate permite observarea modului în care elevul analizează situațiile particulare, nu doar aplică un tipar memorat.	Dacă sunt identificate dificultăți legate de condiții sau de cazuri particulare, profesorul introduce exerciții scurte și focalizate, integrate în activitățile următoare. Achizițiile deja stabilizate nu sunt reluate printr-o lecție distinctă.
Transfer și raționament algoritmic	**D3 — 8 minute, pe hârtie. Urmărirea execuției unui algoritm.** Se oferă un algoritm de numărare sau de însumare aplicat unei succesiuni scurte de valori. Elevii completează un tabel cu valorile variabilelor după fiecare pas și precizează rezultatul final.	Capacitatea de a **urmări sistematic execuția algoritmului**, de a explica modificarea valorilor și de a corela instrucțiunile cu rezultatul obținut. Se observă atât corectitudinea răspunsului, cât și strategia utilizată pentru a-l obține. Nivel: completează tabelul cu sprijin = **De bază**; independent = **Consolidat**; identifică ce se schimbă dacă șirul este vid = **Avansat**.	Dacă urmărirea execuției produce dificultăți, tabelul de evoluție a valorilor este introdus mai devreme și utilizat sistematic ca instrument de înțelegere, testare și depanare. Pentru elevii care demonstrează autonomie, pot fi propuse variante ale problemei sau situații-limită.

DOMENIU DIAGNOSTIC	ÎNTREBARE / SARCINĂ	CE URMĂRIM	DECIZIE POSIBILĂ
Raportarea la învățare și la eroare	Elevii răspund la afirmația „Când programul nu funcționează așa cum mă aștept, de obicei…” și la o întrebare deschisă: „Descrie o situație în care ai întâmpinat o dificultate la realizarea unui program. Cum ai încercat să o rezolvi?”	Modul în care elevul se raportează la eroare, la încercare, la verificare și la solicitarea sprijinului, precum și la experiența sa anterioară de rezolvare a problemelor prin programare. Răspunsurile oferă indicii despre persistență și strategiile de lucru, fără a fi utilizate pentru etichetarea elevului.	Profesorul poate ajusta nivelul de ghidaj, organizarea colaborării, tipul de feedback și rutinele de lucru. În etapa inițială, accentul este pus pe feedback descriptiv și pe orientări concrete pentru pașii următori, astfel încât eroarea să poată fi utilizată ca sursă de informație pentru revizuirea soluției.
Metacogniție și autoreglare	„Când un program nu funcționează așa cum te aștepți, care sunt primele trei lucruri pe care le faci, în ordine?”	Existența unei **strategii de deblocare**: citirea mesajului de eroare, localizarea problemei, verificarea valorilor, testarea pe un caz simplu, consultarea unei resurse sau solicitarea argumentată a sprijinului. Se urmărește capacitatea elevului de a-și organiza propriul proces de depanare.	Profesorul construiește împreună cu elevii o rutină de depanare în câțiva pași, care rămâne disponibilă în laborator și este reutilizată în activitățile ulterioare. Elevii revin periodic asupra acesteia și reflectează asupra strategiilor care i-au ajutat să depășească blocajele.

IMPORTANT Rezultatele diagnozei sunt valorificate în proiectarea activităților din săptămânile 2–6, prin ajustarea sarcinilor, a nivelului de sprijin, a ritmului și a gradului de complexitate. Diagnoza își îndeplinește rolul formativ atunci când dovezile obținute fundamentează decizii didactice concrete privind remedierea, consolidarea, diferențierea sau aprofundarea învățării.
În această etapă, standardele de evaluare de la gimnaziu sunt mai utile ca repere pentru interpretarea achizițiilor-ancoră, iar cele trei sarcini trebuie citite împreună pentru a decide de ce sprijin are nevoie elevul și care este următorul pas în învățare, nu pentru a-i atribui un nivel global.

6.Reflecție asupra proiectării unității de învățare

Fișa sprijină profesorul în proiectarea coerentă a unității de învățare, pornind de la competențele vizate, achizițiile anterioare relevante și dovezile de învățare, și în ajustarea demersului pe baza progresului observat. Exemplele sunt orientative și se adaptează programei, timpului alocat disciplinei și particularităților clasei.

CÂMP	ÎNTREBARE DE PROIECTARE	DE COMPLETAT
Ideea centrală a unității de învățare	Care este înțelegerea esențială pe care elevul o construiește în această unitate de învățare?	O soluție informatică se **construiește progresiv**, printr-un proces care presupune analiza problemei, proiectarea soluției, implementarea, testarea și îmbunătățirea acesteia. Scrierea codului este o etapă a procesului, nu punctul de plecare.
Elementul curricular dezvoltat	Ce element nou al programei devine vizibil în această unitate de învățare?	Domeniul *Strategii de rezolvare a problemelor*, conținutul 2.1 — **Principii de elaborare a unui program**: prin valorificarea principiilor de elaborare a unui program: gândire computațională, analiza problemei, proiectarea și reprezentarea algoritmului, implementarea, testarea și depanarea, proiectarea modulară și, în limitele programei specifice specializării, analiza eficienței soluțiilor.
Competențe specifice vizate	Ce competențe sunt dezvoltate efectiv prin activitățile propuse?	Sunt selectate din programa specifică specializării competențele care vizează identificarea și explicarea strategiilor de rezolvare, implementarea soluțiilor, precum și analiza și evaluarea corectitudinii și eficienței acestora. Se evită asocierea formală a unui număr mare de competențe cu aceeași activitate; sunt urmărite prioritar cele pentru care pot fi colectate dovezi relevante. Astfel: **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. **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· **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· **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 **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
Achiziții anterioare relevante	Care sunt achizițiile formate în gimnaziu pe care se construiesc noile învățări?	Noțiunea de algoritm, identificarea datelor de intrare și de ieșire, utilizarea variabilelor și expresiilor, structurile secvențiale, alternative și repetitive, algoritmii elementari de prelucrare și experiența anterioară de implementare, testare și depanare. Nivelul real al acestor achiziții este evidențiat prin sarcinile de diagnostic integrate în primele activități. Nivelul efectiv se stabilește prin sarcinile D1–D3 din săptămâna 1.
Întrebări esențiale	Ce întrebări pot orienta învățarea și pot fi reluate pe parcursul unității?	„Cum știu că soluția mea este corectă și nu doar că funcționează pentru exemplul testat?” · „Cum mă ajută descompunerea unei probleme să construiesc o soluție mai clară?” · „După ce criterii pot compara două soluții pentru aceeași problemă?”

Dovezi ale învățării	Ce va face sau ce va produce elevul pentru a demonstra progresul?	În funcție de activitățile selectate: fișa de analiză a problemei, reprezentarea algoritmului, programul implementat, schema modulară, setul de teste, explicația strategiei, jurnalul de depanare, revizuirea unei soluții și compararea argumentată a unor alternative. Se urmăresc atât produsele finale cât și dovezile intermediare ale procesului de rezolvare.
Criterii de reușită (descriptori)	Cum descriu reușita în realizarea sarcinii și progresul elevului?	Criteriile sunt formulate în termeni observabili și accesibili elevului: identifică datele și cerințele problemei; explică strategia aleasă; proiectează o soluție coerentă; implementează algoritmul; construiește teste relevante; identifică și corectează erorile; justifică rezultatul și, după caz, compară soluțiile. Gradul de autonomie, complexitatea sarcinii și calitatea argumentării pot diferenția nivelurile de performanță.
Secvențe de învățare	Cum valorific achizițiile anterioare și cum construiesc progresiv autonomia și transferul?	Săptămânile 1–5: L1 ciclul complet → L2 cele trei reprezentări → L3 testarea → L4 proiectarea modulară → L5 eficiența. Activitățile urmăresc o progresie de tip **reactivare și diagnoză → modelare și proiectare → implementare → testare și depanare → analiză și îmbunătățire → transfer**. La specializarea **Matematică-informatică**, cele 2 ore/săptămână permit aprofundarea și alternarea activităților de conceptualizare și de laborator.
Evaluare formativă	Ce dovezi colectez pe parcurs și cum le folosesc pentru a regla învățarea?	Pot fi utilizate bilete de ieșire, observarea sistematică, întrebări de reflecție, analiza codului și a seturilor de teste, autoevaluarea, evaluarea între colegi și feedbackul descriptiv. Feedbackul este oferit suficient de devreme pentru ca elevul să poată revizui și să îmbunătățească soluția înainte de evaluarea finală. Diagnoza se poate obține: - prin Bilet de ieșire (instrumentul 8.1) - observare pe grilă în ora practică - feedback scris pe setul de teste, înainte de finalizarea programului, cu revizuire obligatorie.
Diferențiere	Ce pot adapta fără a pierde din vedere competența urmărită?	Se pot varia nivelul de sprijin, complexitatea problemei, reprezentarea datelor, numărul și natura restricțiilor, resursele oferite, gradul de autonomie, organizarea colaborării și ritmul de lucru. De exemplu, un elev poate primi o fișă cu pași sau un fragment de cod incomplet, iar altul poate primi o cerință deschisă sau o problemă care permite compararea mai multor strategii. Competența vizată rămâne reperul comun.

Resurse	Ce resurse educaționale pot fi utilizate pentru a completa manualul care susține învățarea?	Medii de programare adecvate contextului școlii, instrumente pentru vizualizarea execuției, seturi de date, exemple și contraexemple, fragmente de cod, fișe de laborator și instrumente de evaluare. Mediu de dezvoltare dintre cele indicate de programă: • **Medii de dezvoltare pentru Python și C++**, selectate în funcție de limbajul utilizat și de infrastructura disponibilă: IDLE, Thonny și PyCharm pentru Python; Visual Studio Code, configurat corespunzător, pentru Python și C++;, CodeBlocks • **Medii și compilatoare online**, precum OnlineGDB, Replit și Programiz, care pot fi utilizate pentru scrierea, executarea și testarea programelor în **Python și/sau C++**, în funcție de funcționalitățile disponibile. • **Instrumente pentru vizualizarea execuției programelor**, precum Python Tutor, utilizate pentru urmărirea pas cu pas a execuției și a modificărilor valorilor variabilelor, în limbajele suportate. • **Instrumente pentru vizualizarea și explorarea algoritmilor**, precum VisuAlgo și Algorithm Visualizer, utile pentru înțelegerea funcționării structurilor de date și a algoritmilor, inclusiva **metodelor de căutare și de sortare**. • Fișe scurte de laborator, cu enunț, criterii și date de test. Resursele digitale sunt selectate în funcție de scopul didactic, verificate înainte de utilizare și folosite cu respectarea condițiilor de acces și de reutilizare.
Valorificarea rezervei de 25%	Ce decizii privind remedierea, consolidarea, aprofundarea sau extinderea rezultă din dovezile de învățare?	Destinația rezervei nu este stabilită în prealabil. Profesorul analizează tiparele de eroare, gradul de autonomie, calitatea produselor, rezultatele evaluării formative și dificultățile recurente și decide intervenția adecvată: **remediere** pentru achiziții esențiale insuficient formate; **consolidare** pentru stabilizarea și transferul acestora; **aprofundare** pentru dezvoltarea înțelegerii; **extindere** prin probleme noi, strategii alternative sau sarcini cu un grad sporit de complexitate.
Reflecție și decizii ulterioare	Ce voi analiza la finalul unității pentru a proiecta etapa următoare?	Câți elevi au trecut de la „De bază” la „Consolidat” între săptămâna 1 și săptămâna 6? Câți dintre elevi construiesc spontan un caz-limită? Ce enunț a generat cele mai multe întrebări procedurale — semn că fișa de laborator trebuie rescrisă, nu explicată din nou? Ce pot realiza elevii acum cu mai multă autonomie? Ce dificultăți persistă? Ce tipuri de erori se repetă? Ce strategii de testare și depanare au început să utilizeze spontan? Ce sarcini au produs cele mai relevante dovezi de învățare? Ce trebuie ajustat în explicații, resurse, nivelul de sprijin sau în proiectarea următoarei unități?

7.Reflecție asupra activităților de învățare

Activitățile de învățare sunt proiectate astfel încât elevul să activeze achizițiile anterioare, să exploreze problema, să construiască înțelegerea, să aplice noile achiziții, să le transfere în contexte diferite și să reflecteze asupra propriului demers. Momentele de mai jos constituie repere orientative; ele pot fi combinate, reluate sau distribuite în mai multe ore, în funcție de competențele vizate, complexitatea sarcinii și de timpul disponibil.

MOMENT	SCOP	ÎNTREBĂRI DE PROIECTARE	EXEMPLU / DE COMPLETAT
ANCORARE	Activează achizițiile anterioare și creează contextul pentru învățarea nouă.	Ce achiziții pot valorifica elevii? Ce problemă sau situație poate genera interes și poate evidenția nevoia de învățare suplimentară?	Profesorul propune o problemă accesibilă, cu un rezultat ușor de verificat, de exemplu: „Dintr-o serie de note, determinați câte sunt mai mari decât 8 și calculați media acestora.” Elevii descriu inițial, fără cod, cum ar aborda problema. Strategiile propuse devin un punct de plecare pentru analiza și structurarea procesului de rezolvare.
EXPLORARE	Generează observații, ipoteze, întrebări și nevoia de a investiga.	Ce poate investiga, compara, anticipa sau testa elevul? Ce situație îl determină să își verifice propria strategie?	Elevii analizează **două soluții pentru aceeași problemă**, dintre care una produce un rezultat incorect într-o anumită situație. Identifică datele de test care diferențiază soluțiile și formulează explicații. Astfel, necesitatea testării rezultă din investigarea problemei, înainte de formalizarea criteriilor de testare.
CONSTRUIREA ÎNȚELEGERII	Organizează și explicitează conceptele, relațiile și strategiile identificate.	Ce concept, principiu sau procedură trebuie să devină explicită? Cum conectez observațiile elevilor cu limbajul și instrumentele specifice informaticii?	Profesorul sistematizează, împreună cu elevii, procesul de elaborare a soluției: **analiză → proiectare → implementare → testare și depanare → evaluare → îmbunătățire**. Sunt comparate, acolo unde este relevant, diferite modalități de reprezentare a algoritmului, iar fișa de analiză este introdusă ca instrument de sprijin pentru rezolvarea problemelor.
APLICARE	Consolidează achizițiile prin utilizarea lor în situații apropiate de cele explorate.	Cum utilizează elevul noua achiziție? Ce dovezi îmi arată că strategia poate fi aplicată cu un grad progresiv de autonomie?	Elevii implementează în limbajele **Python** și **C++** soluția proiectată și o verifică utilizând propriul set de teste. Activitatea poate fi realizată individual sau colaborativ; în lucrul în perechi, rolurile de proiectare, implementare, testare și verificare pot fi alternate pentru ca fiecare elev să participe la etape diferite ale procesului.

MOMENT	SCOP	ÎNTREBĂRI DE PROIECTARE	EXEMPLU / DE COMPLETAT
TRANSFER	Valorifică achizițiile într-un context nou și dezvoltă autonomia în rezolvarea problemelor.	Poate elevul să selecteze și să adapteze ceea ce a învățat într-o situație diferită? Poate explica și argumenta alegerile făcute?	Elevul rezolvă o problemă nouă, cu un grad redus de ghidaj, și decide ce strategie, ce reprezentare și ce teste sunt adecvate. De exemplu, procesul exersat anterior este transferat către o problemă de prelucrare a cifrelor unui număr. Profesorul urmărește nu doar obținerea rezultatului, ci și modul în care elevul transferă strategia de rezolvare.
REFLECȚIE	Face vizibile învățarea, dificultățile și strategiile utilizate și fundamentează pașii următori.	Ce a înțeles elevul? Ce strategie a funcționat? Unde a întâmpinat dificultăți? Ce trebuie consolidat sau schimbat în activitatea următoare?	Prin intermediul unui bilet de ieșire, al unei scurte autoevaluări sau al unei întrebări de reflecție, elevul analizează propriul proces: „Ce etapă a procesului de rezolvare mi-a fost cea mai utilă astăzi? „Unde am întâmpinat o dificultate și cum am depășit-o?”, „Ce aș face diferit la următoarea problemă?”. Profesorul corelează aceste răspunsuri cu celelalte dovezi de învățare pentru a ajusta activitățile, nivelul de sprijin și gradul de complexitate al sarcinilor următoare.

8. Instrumente utile

8.1. Verificare rapidă – 3 minute (instrument pentru elev)

PROMPT	RĂSPUNSUL ELEVULUI
Știu / Pot să…	
Încă nu este clar…	
Pentru a merge mai departe, am nevoie de…	

8.2. Feedback pentru învățare (instrument pentru profesor)

Feedbackul pentru învățare pornește de la ceea ce elevul demonstrează în produsul sau activitatea realizată, îl raportează la criteriul de reușită și indică un pas următor concret și realizabil.

1. CE OBSERV – dovada actuală a învățării	2. CE URMĂRIM – criteriul de reușită	3. CE FACI ÎN CONTINUARE – pas concret pentru îmbunătățire
Programul rulează corect pentru exemplul din enunț, iar soluția este clar organizată. Setul de teste include însă numai situații uzuale, astfel încât nu avem încă suficiente dovezi că soluția funcționează și pentru situații particulare sau cazuri-limită.	Următorul pas este să verifici soluția folosind **teste variate și relevante**, inclusiv cazuri-limită, și să explici ce demonstrează fiecare test despre corectitudinea programului.	Adaugă **două cazuri-limită** relevante pentru problema rezolvată. Înainte de a modifica programul, anticipează rezultatul pentru fiecare caz, apoi rulează programul și compară rezultatul obținut cu cel așteptat. Notează ce ai constatat și, dacă este necesar, revizuiește soluția și testeaz-o din nou.

8.3. Listă de verificare pentru integrarea responsabilă a tehnologiilor digitale și a inteligenței artificiale

Înainte de utilizarea unui instrument digital sau bazat pe inteligență artificială, profesorul verifică dacă activitatea contribuie efectiv la dezvoltarea competențelor vizate și permite identificarea contribuției proprii a elevului.

- **Relevanță pentru învățare:** Utilizarea tehnologiei/inteligenței artificiale susține analiza, rezolvarea de probleme și luarea unor decizii sau se limitează la generarea/preluarea unor răspunsuri?
- **Gândire critică și verificare:** Sarcina solicită elevului să verifice, să compare, să testeze și să evalueze critic rezultatele furnizate de inteligența artificială ?
- **Contribuția elevului:** Produsul și procesul de lucru oferă dovezi suficiente pentru a distinge contribuția proprie a elevului de conținutul generat cu ajutorul inteligenței artificiale?
- **Transparența utilizării:** Sunt precizate, înaintea activității, modalitățile în care inteligența artificială poate fi utilizată și modul în care elevul o menționează și o descrie?
- **Validarea rezultatelor:** Elevul verifică, prin surse, exemple, cazuri de test sau alte metode adecvate, corectitudinea rezultatului obținut cu ajutorul inteligenței artificiale ?
- **Protecția datelor:** Activitatea evită introducerea în instrumentele de inteligență artificială a datelor cu caracter personal, confidențiale sau a altor informații care nu trebuie divulgate?
- **Drepturi de autor și utilizare responsabilă:** Sunt respectate condițiile de utilizare a resurselor, atribuirea surselor și drepturile de autor?
- **Valoare adăugată:** Utilizarea tehnologiei sau a inteligenței artificiale aduce o contribuție reală procesului de învățare, comparativ cu realizarea aceleiași sarcini fără acest instrument?

8.4. Listă de verificare pentru sarcini de lucru colaborativ/grupe

Înainte de organizarea unei activități de grup, profesorul verifică dacă sarcina susține atât colaborarea autentică, cât și participarea și învățarea fiecărui elev.

- **Participare individuală**: Sarcina oferă fiecărui elev un rol și o contribuție clară, observabilă și relevantă pentru realizarea rezultatului comun?
- **Interdependență**: Rezolvarea sarcinii presupune valorificarea contribuțiilor tuturor membrilor grupului, astfel încât produsul final să nu poată fi realizat integral de unul sau doi elevi?
- **Interacțiune și argumentare**: Activitatea creează contexte în care elevii își explică strategiile, compară soluții, argumentează alegerile și iau decizii împreună?
- **Responsabilitate individuală**: Profesorul poate identifica, pe parcursul activității, ce face, ce explică și ce demonstrează că a învățat fiecare elev, nu doar calitatea produsului final al grupului?
- **Claritatea sarcinii și a rolurilor**: Elevii cunosc de la început ce au de realizat, cum se organizează colaborarea, care sunt responsabilitățile fiecăruia și ce rezultat este așteptat?
- **Criterii de reușită**: Sunt comunicate, înaintea activității, criterii clare pentru aprecierea atât a produsului comun, cât și, acolo unde este relevant, a contribuției individuale și a procesului de colaborare?
- **Reflecție asupra colaborării**: Activitatea oferă elevilor posibilitatea de a analiza, la final, cum au colaborat, ce contribuție a avut fiecare și ce ar putea îmbunătăți la următoarea activitate de grup?

9. Anexe și resurse de dezvoltat

RESURSĂ	CONȚINUT MINIM	FORMAT RECOMANDAT
Harta schimbării curriculare – clasa a IX-a	Sinteză vizuală a principalelor elemente de continuitate și dezvoltare față de gimnaziu: finalitățile disciplinei, competențele generale și specifice, domeniile de conținut, utilizarea limbajelor **Python** și **C++,** noile contexte și niveluri de complexitate, orientările privind evaluarea și flexibilitatea curriculară. Harta evidențiază distinct particularitățile programelor pentru Matematică-informatică, clase cu predarea Informaticii în regim intensiv	2 pagini / infografic + tabel
Harta de lectură a programei pentru clasa a IX-a	Reprezentare sintetică a relațiilor dintre **competențe generale → competențe specifice → conținuturi → exemple de activități de învățare → dovezi ale învățării.** Include repere pentru interpretarea programei și semnalează câteva abordări de evitat: transformarea listei de conținuturi într-o succesiune rigidă de lecții, tratarea exemplelor de activități ca pe prescripții și aprofundarea unor conținuturi dincolo de nivelul solicitat de programă.	2–3 pagini, vizual
„Cum învățăm informatică la liceu?" – pentru elev	Prezintă, într-un limbaj accesibil, ce se dezvoltă față de gimnaziu, cum este abordată rezolvarea unei probleme, de ce analiza și proiectarea preced implementarea, cum se testează și se îmbunătățește o soluție, ce rol are eroarea în învățare și cum se utilizează responsabil instrumentele digitale și inteligența artificială. Include criterii simple pentru autoevaluarea procesului și a produsului realizat.	1 pagină / fișă pentru elev
Diagnoză pentru debutul clasei a IX-a	Cele trei sarcini D1–D3 (date și variabile, structura alternativă, structura repetitivă), fiecare ancorată într-un standard din anexa nr. 15 la OMEC nr. 4.615/2026, cu repere de interpretare pe cele trei niveluri și cu decizia posibilă pentru fiecare nivel. Durată totală: 18 minute. Fără notă.	editabil + repere de răspuns
Resurse – primele 7 săptămâni	Set flexibil de resurse adaptabile celor două specializări: probleme și sarcini de lucru, fișa de analiză a problemei, exemple și contraexemple, seturi de date și teste, tabel pentru urmărirea valorilor variabilelor, jurnal de depanare, reprezentări comparative ale algoritmilor și resurse digitale verificate.	dosar de resurse editabile/ QR / acces digital

RESURSĂ	CONȚINUT MINIM	FORMAT RECOMANDAT
Instrumente pentru evaluare și feedback	Instrumente utilizabile pe parcursul învățării: criterii de reușită, rubrici descriptive, sarcini de evaluare, bilete de ieșire, grile de observare și de autoevaluare. Sunt incluse modele de feedback descriptiv, însoțite de orientări concrete pentru pașii următori, astfel încât elevul să poată revizui și să îmbunătățească soluția. Evaluarea urmărește atât produsul, cât și procesul de analiză, proiectare, implementare, testare și revizuire.	2–3 pagini + instrumente editabile
Repere pentru valorificarea rezervei de 25%	Exemple de decizii privind utilizarea flexibilității curriculare, fundamentate pe dovezile de învățare colectate. Sunt ilustrate scenarii de remediere, consolidare, aprofundare și extindere, fără a se asocia anticipat rezerva cu o anumită săptămână sau cu praguri numerice rigide. Exemplele arată cum pot fi adaptate sarcina, nivelul de sprijin, complexitatea, contextul și gradul de autonomie în funcție de nevoile identificate.	1–2 pagini / hartă decizională

10. Fișe și resurse de lucru — pachetul de start

Această parte dezvoltă efectiv resursele inventariate în secțiunea 9. Fiecare anexă este gata de multiplicat sau de proiectat în laborator, fără prelucrări suplimentare. Ordinea anexelor urmează ordinea resurselor din tabelul secțiunii 9.

ANEXA	RESURSA DIN TABELUL SECȚIUNII 9	CÂND SE FOLOSEȘTE
A	Harta schimbării curriculare — clasa a IX-a	în ședința de catedră, în prima săptpmână; se afișează în laborator
B	Harta de lectură a programei — matricea 6 × 5, ordinea celor 10 poziții, erori de evitat	la proiectarea unității 1 și ori de câte ori se alege o problemă
C	„Cum învățăm la disciplina Informatică la liceu?” — pentru elev	săptămâna 1, la finalul primei ore; se păstrează în portofoliu
D	Diagnoza pentru start (D1–D3) + grila profesorului	săptămâna 1, ora de activități practice
E	Instrumentele permanente + fișe de laborator	în fiecare oră de activități practice; fișele se distribuie la începutul orei
F	Evaluare și feedback: rubrica, testul din săptămâna 6, instrumentele elevului	criteriile se anunță în săptămâna 2; testul se aplică în săptămâna 6
G	Harta deciziilor pentru cei 25%	pe baza dovezilor de învățare colectate, pentru stabilirea activităților de **remediere, consolidare, aprofundare sau extindere** și pentru ajustarea proiectării unității de învățare următoare.

Anexa A. Harta schimbării curriculare — o pagină pentru catedră

Formatul recomandat: o pagină afișabilă. Coloana a treia este singura care contează în practică — o schimbare care nu produce o consecință în ora de luni nu are ce căuta pe această pagină.

DE LA (gimnaziu, Informatică și TIC)	LA – clasa a IX-a, specializarea Matematică-informatică, clase cu predarea Informaticii în regim intensiv	CE ÎNSEAMNĂ ÎN ACTIVITATEA DIDACTICĂ
Disciplina **Informatică și TIC**, cu un buget de timp de **1 oră/săptămână**, urmărește formarea competențelor digitale și algoritmice de bază.	La specializarea Matematică-informatică, clase cu predarea informaticii în regim intensiv, disciplina Informatică beneficiază de un buget de timp extins, de 4 ore/săptămână, care permite aprofundarea competențelor de analiză și modelare a problemelor, proiectare și implementare a soluțiilor, testare și depanare, evaluare a corectitudinii și eficienței, precum și dezvoltarea progresivă a autonomiei elevilor în elaborarea unor soluții informatice cu grad sporit de complexitate.	Cele două ore sunt proiectate **în corelație**, astfel încât înțelegerea conceptelor și proiectarea soluției să fie valorificate prin activități practice de implementare, testare, depanare și îmbunătățire.

DE LA (gimnaziu, Informatică și TIC)	LA – clasa a IX-a, specializarea Matematică-informatică, clase cu predarea Informaticii în regim intensiv	CE ÎNSEAMNĂ ÎN ACTIVITATEA DIDACTICĂ
Competențele generale din gimnaziu vizează utilizarea responsabilă a tehnologiei, rezolvarea unor probleme prin metode specifice informaticii și realizarea unor produse digitale.	Șase competențe generale ale taxonomiei Bloom revizuită: **identifică**, **explică**, **utilizează**, **analizează**, **evaluează**, **elaborează** — în raport cu modelele conceptuale și operaționale specifice dezvoltării soluțiilor informatice.	Profesorul pornește de la **competența vizată**, selectând probleme și activități care permit dezvoltarea acesteia în timpul disponibil. Accentul cade pe înțelegerea și utilizarea funcțională a conceptelor, nu pe acumularea unui număr mare de tehnici.
Competențele specifice construiesc progresiv achizițiile algoritmice și de programare corespunzătoare nivelului gimnazial.	Treizeci de competențe specifice: șase competențe generale × cinci fire tematice (Modele de organizare a datelor; Strategii de rezolvare a problemelor -algoritmi specializați pe clase de probleme; Strategii de rezolvare a problemelor – *principii de elaborare a unui program;- Memorarea datelor și organizarea codului în limbaj de programare* – elemente de memorare a datelor; *Memorarea datelor și organizarea codului în limbaj de programare* – elemente de organizare a codului).	Programa este citită prin relațiile dintre competențe, domenii de conținut și activități de învățare, nu ca o succesiune de teme care trebuie parcurse mecanic. Profesorul proiectează progresiv sarcini care solicită niveluri tot mai ridicate de autonomie și de raționament. Programa se citește pe competențe și pe fire tematice, nu pe lista de conținuturi.
Elevii pot avea experiențe diferite de reprezentare și implementare a algoritmilor, inclusiv prin intermediul unor medii grafice sau al unor limbaje de programare folosite în gimnaziu.	Programa prevede explicit **armonizarea limbajelor la trecerea în clasa a IX-a**: toți elevii utilizează limbajul precizat în programa în vigoare, Python. Iar la clasele cu predarea intensivă a disciplinei Informatică se utilizează și limbajul C++.	Profesorul identifică experiențele anterioare de programare ale elevilor și, atunci când este necesar, facilitează transferul dintre reprezentările și limbajele deja cunoscute și cele utilizate în clasa a IX-a. Accentul se pune pe valorificarea și transferul gândirii algoritmice, nu pe reluarea sistematică a noțiunilor elementare de algoritmică. Mediile de programare pentru Python și C++ sunt configurate și verificate înainte de activitățile practice, astfel încât timpul de laborator să fie valorificat pentru învățare.
Elevii dobândesc noțiuni fundamentale despre algoritmi, date, structuri de control și rezolvarea unor probleme accesibile.	Conținuturile clasei a IX-a extind aceste achiziții către modelarea și organizarea datelor, strategiile de rezolvare a problemelor, proiectarea modulară, subprogramele, memorarea și prelucrarea datelor, precum și analiza corectitudinii și eficienței soluțiilor, în limitele prevăzute de programă.	Conținuturile sunt tratate ca resurse pentru dezvoltarea competențelor, nu ca o listă de lecții. Profesorul le selectează și le organizează în situații de învățare care solicită elevului să analizeze, să proiecteze, să implementeze, să testeze și să își justifice soluția.

DE LA (gimnaziu, Informatică și TIC)	LA – clasa a IX-a, specializarea Matematică-informatică, clase cu predarea Informaticii în regim intensiv	CE ÎNSEAMNĂ ÎN ACTIVITATEA DIDACTICĂ
Evaluarea valorifică produsele și performanțele realizate de elev în raport cu competențele urmărite în gimnaziu.	În clasa a IX-a se consolidează rolul evaluării formative, prin colectarea de dovezi pe parcursul procesului de învățare, alături de evaluarea sumativă. Sunt urmărite nu numai rezultatele obținute, ci și strategiile, testarea, depanarea, argumentarea și revizuirea soluțiilor.	Criteriile de reușită sunt comunicate odată cu sarcina și sunt utilizate pentru feedback, autoevaluare și revizuirea produsului. Profesorul colectează dovezi atât despre produsul realizat, cât și despre procesul prin care elevul a ajuns la soluție.
Timpul didactic este organizat în raport cu competențele și conținuturile prevăzute pentru gimnaziu.	Programa de liceu oferă flexibilitate curriculară prin rezerva de 25% din bugetul de timp, care poate fi valorificată în funcție de nevoile de învățare identificate.	Destinația rezervei nu este stabilită rigid la începutul anului. Pe baza diagnozei și a dovezilor colectate pe parcurs, profesorul decide utilizarea acestora pentru remediere, consolidare, aprofundare sau extindere și reflectă această decizie în proiectarea didactică.

Activitățile practice se desfășoară în laboratorul de informatică, în condiții care permit fiecărui elev accesul la un calculator și utilizarea aplicațiilor necesare pentru realizarea sarcinilor de lucru. Organizarea și configurarea spațiului și a echipamentelor trebuie să susțină atât activitatea individuală la calculator, cât și demonstrațiile, explicațiile și interacțiunea didactică.

Anexa B. Harta de lectură a programei

Cele 30 de competențe specifice ale clasei a IX-a se citesc pe două axe. Pe **verticală** — progresia taxonomică în interiorul unui fir tematic: aceeași temă, de la identificare la elaborare. Pe **orizontală** — același nivel de exigență, aplicat la 5 conținuturi diferite.

FIRUL TEMATIC (coloana x.n)	CE URMĂREȘTE	CELE ȘASE TREPTE, DE LA CS 1.x LA CS 6.x
x.1 — Organizarea datelor modele conceptuale fundamentale: date simple sau liste	Ce informații trebuie prelucrate și cum sunt legate între ele — înainte de a decide ce tip de variabilă să folosești. Include stiva, coada, lista cu acces direct sau secvențial, lista de frecvențe.	**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 **2.1** Explică principiile de organizare a datelor în cadrul unor modele conceptuale fundamentale - date simple sau liste, pentru a le gestiona eficient **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 **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 **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 **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
x.2 — Algoritmi specializați pe clase de probleme: numere, sortare, generare sistematică	Repertoriul de algoritmi al clasei a IX-a: prelucrări ale datelor numerice (de exemplu, prelucrarea cifrelor, divizori, algoritmul lui Euclid, baze de numerație), generare sistematică a elementelor unei liste, metode de sortare (prin selecția minimului, listă de frecvențe, metoda bulelor).	**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 **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· **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 **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· **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· **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

FIRUL TEMATIC (coloana x.n)	CE URMĂREȘTE	CELE ȘASE TREPTE, DE LA CS 1.x LA CS 6.x
x.3 — Proiectarea modulară strategii de rezolvare prin descompunere	Descompunerea unei probleme în subprobleme și organizarea rezolvării pe module — firul care structurează unitatea 1 și face posibilă trecerea la subprograme.	**1.3** Identifică specificul, caracteristicile și etapele unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor **2.3** Explică etapele unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor·. **3.3** Aplică strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor· **4.3** Analizează aplicabilitatea, avantajele și dezavantajele utilizării unor strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor· **5.3** Evaluează corectitudinea și eficiența algoritmilor care utilizează strategii de rezolvare a problemelor prin proiectarea modulară a algoritmilor· **6.3** Proiectează modular algoritmi personalizați pentru rezolvarea problemelor
x.4 — Elemente de limbaj pentru prelucrarea datelor și transmiterea lor la și de la program	Clasa list din Python, operatorii și metodele ei; fișierele text; obiectele grafice de bază. Aici intră și „a citi și a afișa”, adică interfața programului cu exteriorul.	**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 **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 **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 **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·. **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· **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

FIRUL TEMATIC (coloana x.n)	CE URMĂREȘTE	CELE ȘASE TREPTE, DE LA CS 1.x LA CS 6.x
x.5 — Subprograme definiție, apel, subprograme predefinite	Antet, corp, parametri, variabile locale și globale, returnarea rezultatelor, mecanismul de executare; subprogramele predefinite în limbajele Python și C++ pentru calcule uzuale și pentru colecții.	**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· **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· **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· **4.5** Analizează aplicabilitatea, avantajele și dezavantajele utilizării subprogramelor, în implementarea algoritmilor în limbaj de programare· **5.5** Evaluează corectitudinea și eficiența programelor care utilizează subprograme, pentru a rezolva probleme informatice·. **6.5** Implementează aplicații în limbaj de programare care integrează subprograme personalizate, pentru a modulariza și organiza eficient codul în cadrul soluțiilor informatice

În primele 7 săptămâni se parcurg treptele 1–4 pe firul x.3, cu intrare în x.1 (fișa de analiză a datelor) și în x.2 (algoritmii de bază, în săptămâna 7). Treptele 5 și 6 — evaluarea și elaborarea — se ating în unitățile următoare, pe aceleași fire.

B.1. Ordinea de abordare recomandată de programă (cele zece poziții)

Programa oferă un exemplu orientativ de ordine a conținuturilor pentru clasa a IX-a, specializarea matematică-informatică. Primele șapte săptămâni acoperă integral pozițiile 1–2 și deschid poziția 3.

NR.	CONȚINUTUL, ÎN FORMULAREA PROGRAMEI	UNDE SE AFLĂ ÎN CELE 7 SĂPTĂMÂNI
1	Strategii de rezolvare a problemelor. Principii de elaborare a unui program	Săptămâna 1 – unitatea 1
2	Strategii de rezolvare a problemelor. Prelucrări ale datelor numerice	Săptămânile 2-7— unitatea 2
3	Memorarea datelor și organizarea codului în limbaj de programare. Subprograme	
4	Memorarea datelor și organizarea codului. Introducere în programarea orientată pe obiecte	
5	Memorarea datelor și organizarea codului. Biblioteca Tkinter din Python pentru interfețe grafice	
6	Memorarea datelor și organizarea codului. Fișiere text	
7	Organizarea conceptuală a datelor. Modelul conceptual liniar	
8	Memorarea datelor și organizarea codului. 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	

B.2. Trei erori de interpretare a programei, de evitat

EROAREA	DE CE APARE	CE FAC ÎN LOC
Lista de conținuturi devine un plan de lecții. Se predau, în ordinea din tabel, „modelul liniar”, apoi „prelucrări numerice”, apoi „subprograme”.	Tabelul conținuturilor este singura parte a programei care arată ca o programă veche și se citește automat ca succesiune.	Pornește de la competența specifică vizată și alege conținutul care o susține. Ordinea recomandată de programă (B.1) nu coincide cu ordinea din tabelul conținuturilor și este cea operațională.
Exemplele de activități de învățare sunt obligatorii. Se parcurg toate ca o listă de exerciții de rezolvat.	Sunt numeroase, formulate detaliat și par a fi un barem.	Programa precizează că are un rol orientativ, nu prescriptiv: poate fi adaptată, completată sau înlocuită. Aleg două-trei pe competență; restul rămân rezervoare de probleme.
Tkinter și programarea orientată pe obiecte se predau ca teme de aprofundare. Se ajunge la moștenire, la clase proprii, la evenimente complexe.	Sunt subiecte atractive și par „adevărata programare”.	Programa cere explicit doar noțiunile strict necesare pentru utilizarea, la nivel elementar, a unor clase predefinite: instanțiere, acces la membri, câteva obiecte grafice. Aprofundarea aparține claselor următoare.

CRITERIUL DE VERIFICARE Înainte de proiectarea unei lecții, profesorul își formulează două întrebări esențiale: Ce competență specifică urmăresc să dezvolt? Prin ce acțiune, produs sau rezultat observabil va demonstra elevul că a învățat? Dacă dovezile anticipate se limitează la faptul că elevul „ascultă" sau „notează", proiectarea activității trebuie completată cu sarcini prin care elevul analizează, aplică, construiește, testează, explică sau argumentează.

Anexa C. „Cum învățăm informatică la liceu?” — pagina elevului

Se distribuie în săptămâna 1, în ultimele 10 minute ale orei practice, și se păstrează în portofoliu. Textul este scris pentru elev, la persoana a doua, fără termeni din documentele curriculare.

ÎNTREBAREA TA	RĂSPUNSUL, PE SCURT
Ce se schimbă față de gimnaziu?	În clasa a IX-a vei folosi ceea ce ai învățat deja despre algoritmi și programare pentru a rezolva sistematic probleme mai complexe. Vei acorda mai multă atenție analizei problemei, proiectării soluției, implementării în limbajele Python și C++, testării și îmbunătățirii programului La clasele cu predarea disciplinei informatică în regim intensiv; sunt 4 ore pe săptămână: 2 ore de studiu teoretic și 2 ore de activități practice.
Cum abordez o problemă de informatică?	Nu începi direct prin a scrie cod. Mai întâi înțelegi problema: ce date primești, ce rezultat trebuie să obții, ce condiții trebuie respectate și ce strategie poți folosi. Apoi proiectezi soluția, o implementezi, o testezi și, dacă este necesar, o corectezi și o îmbunătățești.
Cum se vor desfășura activitățile practice?	Primești o problemă sau o sarcină și analizezi cerințele înainte de implementare. Identifici subproblemele/modulele, modelul sau modelele de organizare a datelor, apoi scrii programul în Python, alegi date potrivite pentru testare și verifici dacă rezultatele sunt cele așteptate. Dacă apare o eroare, urmărești execuția, verifici valorile variabilelor și identifici unde trebuie revizuită soluția.
De ce analizez problema înainte de a scrie codul?	Pentru că un program corect pornește de la o înțelegere corectă a problemei. Analiza te ajută să identifici datele, restricțiile și rezultatul urmărit și să alegi o strategie înainte de implementare. Astfel, poți preveni erorile și poți construi o soluție mai clară și mai ușor de verificat.
Când consider că soluția este finalizată?	Nu este suficient ca programul să ruleze pentru exemplul din enunț. Trebuie să poți explica soluția, să o testezi pe date relevante, să verifici rezultatele și să corectezi eventualele erori. În funcție de sarcină, vei documenta și etapele importante ale rezolvării acesteia.
Ce fac atunci când programul nu funcționează așa cum mă aștept?	Tratezi eroarea ca pe o sursă de informație. Citești mesajele afișate, localizezi zona în care apare problema, urmărești valorile importante, testezi pe un caz simplu și revizuiești soluția. Important nu este doar să corectezi eroarea, ci și să înțelegi de ce a apărut și cum ai rezolvat-o.
Pot utiliza inteligența artificială?	Poți utiliza instrumente de inteligență artificială atunci când sarcina și regulile stabilite de profesor permit acest lucru. Menționezi cum le-ai folosit, verifici critic rezultatele și îți asumi deciziile privind soluția finală. Inteligența artificială poate oferi sugestii, dar tu trebuie să înțelegi, să testezi, să verifici și să poți explica soluția pe care o prezinți.
Cum este evaluată învățarea mea?	Pe criteriile pe care le afli înainte de sarcină. Fiecare criteriu are trei niveluri: **De bază** (rezolvi cu sprijin sau după un model), **Consolidat** (rezolvi singur și explici de ce ai ales așa), **Avansat** (compari două soluții, alegi motivat și tratezi cazurile-limită). Primești și o propunere pentru următorul pas, nu doar o notă. Profesorul va urmări nu numai rezultatul final, ci și cum analizezi problema, cum alegi strategia, cum implementezi și testezi soluția, cum identifici și corectezi erorile și cum îți argumentezi alegerile. Feedbackul primit îți va arăta ce ai realizat bine și ce poți face în continuare pentru a progresa.

ÎNTREBAREA TA	RĂSPUNSUL, PE SCURT
Ce pot valorifica din ceea ce am învățat în gimnaziu?	Nu începi de la zero. Vei valorifica ceea ce știi deja despre algoritmi, date și variabile, expresii, structuri alternative și repetitive, precum și implementarea și testarea unor programe. Aceste achiziții sunt reactivate atunci când ai nevoie de ele și sunt dezvoltate în probleme și contexte cu un grad progresiv de complexitate.
Cum devin mai autonom în rezolvarea problemelor?	Să parcurgi procesul complet de rezolvare a unei probleme: **să analizezi → să proiectezi → să implementezi → să testezi și să depanezi → să evaluezi → să îmbunătățești soluția** și să poți explica deciziile pe care le-ai luat.

Anexa D. Diagnoza de start — fișa elevului și grila profesorului

Repere pentru administrarea diagnozei
Cele trei sarcini de diagnoză se integrează în activitatea de învățare din prima săptămână, într-un interval scurt de lucru, fără notare și fără a fi prezentate elevilor ca un test. Ele pot constitui una dintre primele activități ale anului și urmăresc identificarea achizițiilor relevante pentru noile învățări. Rezultatele nu sunt utilizate pentru etichetarea sau încadrarea elevilor în niveluri, ci sunt analizate ca dovezi pentru fundamentarea deciziilor didactice privind nivelul de sprijin, diferențierea, ritmul și complexitatea sarcinilor din săptămânile următoare.

D.1. Fișa elevului (se multiplică sau se distribuie digital)

Numele: …… **Clasa:** …… **Data:** ……

Sarcina 1 — date și variabile (pe hârtie). O aplicație primește notele obținute de elevii unei clase la un concurs și trebuie să afișeze câți elevi au peste 8 și media acestora.

CARE SUNT DATELE DE INTRARE?	CARE SUNT DATELE DE IEȘIRE?	DE CE ALTE DATE AVEȚI NEVOIE PE PARCURS?, JUSTIFICAȚI RĂSPUNSUL
……	……	……

Sarcina 2 — structura alternativă. Fie secvența algoritmică de mai jos, dată în limbaj natural, în care se citesc două numere distincte.

```
1.citește a, b
Dacă a>b atunci scrie a, altfel scrie b
```

Scrieți în limbajul de programare studiat algoritmul de mai sus……

Sarcina 3 — structura repetitivă. Fie secvența C++ de mai jos (*), în care toate variabilele sunt întregi. Se citesc numerele 4, 9, 2, 9, 6.

```
n = 5;  s = 0;  k = 0;
for( i = 1; i<= n; i++)
{   cin>>x;
    if( x > 5)
   { k =k + 1;  s =s + x;}
}
cout<<k<<' '<<s;
```

Ce se afișează în urma executării secvenței? …… Ce s-ar întâmpla dacă șirul ar fi vid (n = 0)?……

(*) Se adaptează limbajul, în funcție de limbajele studiate la gimnaziu.

D.2. Grila profesorului — repere de răspuns, interpretare și decizie

SARCINA	RĂSPUNSUL AȘTEPTAT	INTERPRETARE PE TREI NIVELURI	DECIZIA MEA
D1 — date și variabile (ancoră, gimnaziu)	Intrare: numărul de elevi și notele. Ieșire: numărul elevilor cu nota peste 8 și media acestora. Date de manevră: o sumă (eventual nota curentă). Eroarea tipică: se trec ca „date de ieșire" pașii de rezolvare („se compară cu 8").	**De bază:** identifică datele de intrare și de ieșire, cu sprijin. **Consolidat:** identifică independent datele de intrare, de ieșire și de manevră. **Avansat:** justifică de ce contorul și suma trebuie calculate pe parcurs.	Dacă rezultatele diagnozei indică dificultăți frecvente în atingerea reperelor de bază, fișa de analiză a problemei se completează inițial împreună cu întreaga clasă, pe baza unui model oferit de profesor. Dacă achizițiile necesare sunt suficient de stabile, activitatea poate fi organizată direct în perechi, cu un grad mai ridicat de autonomie.
D2 — structura alternativă (ancoră: gimnaziu)	Program complet scris în limbaj de programare.	**De bază:** scrie instrucțiunile, dar incomplet . **Consolidat:** scrie instrucțiunile, dar cu erori minore de sintaxă. **Avansat:** scrie program conform cerinței.	Dacă diagnoza evidențiază dificultăți, se introduc în săptămânile următoare exerciții scurte și focalizate pe activitatea de transfer între reprezentarea în limbaj natural și limbajele de programare Python și C++.
D3 — structura repetitivă (ancore: gimnaziu)	Afișează 3 și 24. Pentru n = 0, ciclul nu se execută: se afișează 0 și 0 — răspuns corect, nu „eroare".	**De bază:** determină doar unul dintre rezultate, cu sprijin. **Consolidat:** determină independent cele două valori. **Avansat:** determină independent cele două valori și tratează corect cazul șirului vid.	Dacă rezultatele diagnozei evidențiază **dificultăți frecvente în urmărirea executării algoritmilor**, la predarea tabelului de evoluție a valorilor variabilelor se utilizează sistematic, pentru o perioadă, ca instrument de înțelegere, verificare și depanare.
Atitudini / motivație	Elevii răspund la câteva afirmații pe o scală de la 1 la 4, referitoare la modul în care reacționează atunci când întâmpină dificultăți în programare. Se adaugă o întrebare deschisă: „Descrie un program sau o problemă de programare care ți s-a părut dificilă. Ce dificultăți ai întâmpinat și cum ai încercat să le depășești?"	Răspunsurile sunt analizate la nivelul clasei pentru a identifica modul în care elevii se raportează la eroare și dificultate, strategiile pe care le folosesc pentru a depăși blocajele și diversitatea experiențelor anterioare de programare. Nu se urmărește etichetarea elevilor, ci identificarea nevoilor de sprijin și a rutinelor de lucru care trebuie dezvoltate.	Profesorul poate adapta organizarea colaborării, nivelul de sprijin și tipul de feedback. În primele activități pot fi utilizate perechi cu niveluri diferite de experiență și roluri clare, cu rotație. Feedbackul este centrat pe procesul de rezolvare, pe strategiile utilizate și pe pașii concreți pentru progres, iar deciziile privind evaluarea produselor sunt stabilite în raport cu scopul și criteriile activității.

SARCINA	RĂSPUNSUL AȘTEPTAT	INTERPRETARE PE TREI NIVELURI	DECIZIA MEA
Metacogniție și autoreglare – activitate orală scurtă	„Atunci când un program nu funcționează așa cum te aștepți, care sunt primele trei lucruri pe care le faci, în ordine?”	Se urmărește dacă elevul dispune de o strategie de deblocare și depanare: identifică problema, verifică informațiile disponibile, testează ipoteze și solicită sprijin în mod adecvat. Răspunsurile oferă indicii despre capacitatea elevului de a-și organiza și de a-și regla propriul proces de rezolvare.	Pornind de la răspunsurile elevilor, clasa construiește, în prima săptămână, o rutină comună de depanare, formulată în câțiva pași clari. Aceasta rămâne disponibilă în laborator și este reutilizată și ajustată pe parcursul activităților de programare, inclusiv în jurnalul de depanare.

IMPORTANT **Diagnoza fundamentează decizii didactice, nu etichetează elevii.** Valoarea ei constă în modul în care informațiile obținute sunt utilizate pentru ajustarea activităților ulterioare. La finalul diagnozei, profesorul trebuie să poată identifica ce se va adapta în săptămânile următoare în ceea ce privește nivelul de sprijin, complexitatea sarcinilor, strategiile de lucru, resursele utilizate sau modalitățile de feedback. În absența unor astfel de ajustări, diagnoza nu își îndeplinește funcția de sprijinire a învățării.

D.3. Fișa de observare a clasei (se completează în timpul orei)

SARCINA	DE BAZĂ (nr. elevi)	CONSOLIDAT (nr. elevi)	AVANSAT (nr. elevi)
D1 — date și variabile	……	……	……
D2 — structura alternativă	……	……	……
D3 — structura repetitivă	……	……	……
Decizia mea pentru săptămâna 2			
Elevii cu care lucrez suplimentar			

Anexa E. Instrumentele permanente și exemple de fișe de lucru

Instrumentele din E.0 au un caracter transversal și pot fi utilizate sistematic în activitățile practice, indiferent de problema abordată.
Fișele E.1–E.10 oferă suport pentru activitățile de laborator din primele șapte săptămâni și includ variante adaptabile pentru remediere, consolidare și extindere, care pot fi valorificate în cadrul timpului aferent rezervei de 25%, în funcție de nevoile de învățare identificate.

E.0.1. Fișa de analiză a problemei (instrument permanent)

Fișa sprijină elevul în înțelegerea și structurarea problemei înainte de implementare și poate fi utilizată sau adaptată în funcție de complexitatea sarcinii și de nivelul de autonomie al elevului.

CÂMP	CE NOTEZ	EXEMPLU COMPLETAT (problema din săptămâna 1)
Problema, în cuvintele mele	Reformulez pe scurt ceea ce trebuie rezolvat, evidențiind rezultatul urmărit.	„Pentru notele elevilor dintr-o clasă, determin câți elevi au obținut o notă mai mare de 8 și calculez media acestor note.”
Date de intrare	Identific datele pe care le primește programul și, atunci când este relevant, precizez tipul, domeniul de valori și modul de furnizare a acestora.	n – numărul elevilor; apoi n note, cu valori între 1 și 10, introduse de la tastatură.
Date de ieșire	Precizez rezultatele pe care trebuie să le furnizeze programul și forma în care acestea vor fi prezentate.	Două valori: numărul elevilor cu nota > 8 și media lor, cu 2 zecimale.
Date necesare pe parcurs	Identific valorile intermediare necesare rezolvării și explic rolul acestora.	s – suma notelor mai mari de 8. Nota curentă este utilizată în momentul citirii și nu trebuie păstrată după prelucrare.
Restricții și situații particulare	Identific valorile admise, condițiile problemei și situațiile care necesită tratare distinctă.	$1 \le n \le 30$, iar notele sunt mai mari sau egale cu 1 și mai mici sau egale cu 10. Dacă nu există nicio notă mai mare decât 8 ($k = 0$), media nu poate fi calculată; programul va afișa un mesaj corespunzător.
Strategia de rezolvare	Descriu, în limbaj natural, etapele esențiale ale soluției, fără a scrie încă programul.	1. Citesc numărul elevilor. 2. Inițializez numărul și suma notelor selectate. 3. Pentru fiecare notă, verific dacă este mai mare decât 8 și, dacă da, actualizez numărul și suma. 4. Dacă există cel puțin o astfel de notă, calculez și afișez media; în caz contrar, afișez un mesaj adecvat.
Cum verific soluția	Construiesc câteva cazuri de test relevante și stabilesc, înainte de rulare, rezultatele așteptate. Includ, atunci când este posibil, un caz uzual și un caz-limită.	$(5\ 9\ 7\ 10\ 8\ 9) \rightarrow 3$ și $9.33 \cdot (3\ 5\ 6\ 7) \rightarrow$ mesaj $\cdot (1\ 10) \rightarrow 1$ și 10.00

E.0.2. Tabelul de evoluție a valorilor (instrument permanent de depanare)

Se completează manual, pe hârtie, înainte de a rula programul — apoi se compară cu execuția reală. Prima linie în care cele două nu coincid este linia cu eroarea.

PASUL	INSTRUCȚIUNEA EXECUTATĂ	VARIABILA 1	VARIABILA 2	CE SE AFIȘEAZĂ
0		……	……	—
1	……	……	……	……
2	……	……	……	……
3	……	……	……	……
…	……	……	……	……

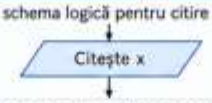
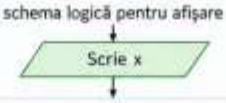
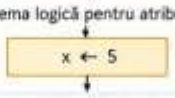
E.0.3. Jurnalul de depanare (se predă odată cu programul)

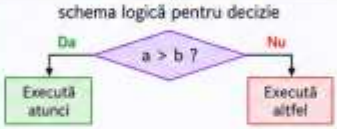
MESAJUL	CE AM PRESUPUS	CE ERA, DE FAPT	CUM AM VERIFICAT
……	……	……	……
……	……	……	……
……	……	……	……

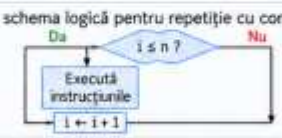
Accentul evaluării cade pe *procesul de identificare, analiză și corectare a erorilor*, nu pe absența lor. Jurnalul de depanare oferă dovezi despre modul în care elevul își verifică soluția, formulează ipoteze, localizează o eroare și o corectează. Identificarea autonomă a unei erori de logică și explicarea demersului de remediere reprezintă o dovadă relevantă de învățare, chiar dacă soluția inițială nu a fost corectă.

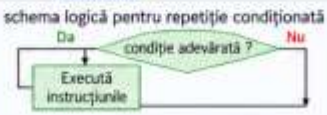
E.0.4. Tabelul comparativ: bloc grafic – pseudocod – Python - C++

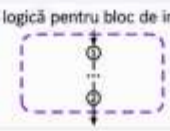
Se completează în primele lecții și rămâne în portofoliu ca punte între gimnaziu și liceu.

CE FACE	SCHEMĂ LOGICĂ (gimnaziu)	PSEUDOCOD	PYTHON	C++
citirea unei valori		*citește* x	`x = int(input())`	`int x;` `cin >> x;`
afișarea unei valori		*scrie* x	`print(x)`	`cout << x;`
atribuirea unei valori		x ← 5	`x = 5`	`int x = 5;`

decizia		**dacă** a > b **atunci** ...
		altfel ...
			`if a > b:`
			`...`
			`else:`
			`...`
				`if (a > b) {`
				`...`
				`} else {`
				`...`
				`}`

repetiția cu contor		**pentru** *i* de la 1 la n
		execută ...
			`for i in range(1, n+1):`
			`...`
				`for (int i = 1; i <= n; i++) {`
				`...`
				`}`

repetiția condiționată		**cât timp** *condiție*
		execută ...
			while *condiție*:
			...
				while (*condiție*) {
				...
				}

blocul de instrucțiuni		indentare sau delimitatori	**indentarea** este obligatorie și are valoare sintactică	se folosesc acolade { } pentru delimitare

UNDE APAR ERORILE DE TRANSFER

În trecerea de la reprezentările și limbajele utilizate anterior la limbajele Python și C++, pot apărea dificultăți recurente care merită **anticipate și explicate explicit**:

Dificultăți de transfer către Python

- **Indentarea** – dacă, în schemele logice sau în mediile cu blocuri, organizarea instrucțiunilor este evidențiată vizual, în Python indentarea delimitează blocurile de instrucțiuni și are rol sintactic.
- **Citirea datelor** – funcția input() returnează un șir de caractere; atunci când problema presupune prelucrarea unor valori numerice, este necesară conversia la tipul de date corespunzător, de exemplu, int(input()).
- Intervalul generat de range() – elevul trebuie să înțeleagă semnificația limitelor intervalului și faptul că valoarea finală nu este inclusă. De exemplu, pentru valorile de la 1 la n se utilizează range(1, n+1).

Aceste particularități sunt abordate preventiv, în contextul primelor activități de programare în Python, prin exemple și comparații cu reprezentările deja cunoscute de elevi, astfel încât dificultățile de transfer să fie transformate în oportunități de înțelegere.

Dificultăți de transfer către C++

În trecerea de la reprezentările și limbajele utilizate anterior la **C++**, pot apărea dificultăți recurente care este util să fie **anticipate și abordate explicit**:

- **Delimitarea blocurilor de instrucțiuni** – dacă, în schemele logice sau în mediile de programare cu blocuri, organizarea instrucțiunilor este evidențiată vizual, în C++ blocurile sunt delimitate prin **acolade { }**. Indentarea contribuie la lizibilitatea codului, dar nu are rolul sintactic pe care îl are în Python.
- **Citirea datelor și declararea tipurilor** – înainte de utilizare, variabilele sunt declarate cu un **tip de date** corespunzător, iar citirea se realizează, de regulă, cu cin. Elevul trebuie să coreleze natura datelor problemei cu tipul ales, de exemplu, int, double sau char.
- **Structura instrucțiunilor și utilizarea separatorilor** – în C++, multe instrucțiuni se încheie cu **punct și virgulă (;)**, iar structurile de control au o sintaxă specifică. Elevii trebuie să distingă între logica algoritmului și convențiile sintactice necesare pentru implementarea acestuia.
- **Repetiția cu contor** – spre deosebire de utilizarea range() în Python, în C++ inițializarea, condiția de continuare și actualizarea contorului sunt precizate explicit în instrucțiunea for. De exemplu, pentru parcurgerea valorilor de la 1 la n se poate utiliza for (int i = 1; i <= n; i++).

Aceste particularități sunt abordate **în contextul primelor activități de programare în C++**, prin exemple comparative și prin raportare la reprezentările și achizițiile deja cunoscute de elevi. În acest mod, atenția rămâne asupra **transferului gândirii algoritmice**, iar diferențele de sintaxă și de organizare a codului devin oportunități de a înțelege relația dintre **algoritm și implementarea acestuia într-un limbaj de programare**.

E.0.5. Mediile de dezvoltare și instrumente digitale pentru activitățile de informatică

Selectarea instrumentelor digitale se realizează în funcție de competențele vizate, scopul activității, nivelul elevilor și infrastructura disponibilă, evitând utilizarea tehnologiei fără o contribuție clară la procesul de învățare.

INSTRUMENT / RESURSĂ	UTILIZARE DIDACTICĂ	REPERE PENTRU UTILIZARE
Medii integrate de dezvoltare (IDE) pentru limbajele Python,și C++ instalate local	Scrierea, rularea, testarea și depanarea programelor; familiarizarea cu funcționalitățile de bază ale unui mediu de dezvoltare.	Programa cere medii integrate de dezvoltare și funcționalități ale acestora. Mediul este configurat și verificat înainte de activitățile practice, astfel încât timpul didactic să fie utilizat pentru rezolvarea problemelor și pentru programare. Se introduc progresiv numai funcționalitățile necesare pentru sarcinile realizate de elevi.
Mediu online pentru programare în Python și C++	O alternativă pentru scrierea și rularea programelor atunci când accesul la mediul local este limitat sau apar dificultăți tehnice.	Este utilă stabilirea unei soluții comune, ușor accesibilă, pentru a asigura continuitatea activității și a reduce timpul necesar pentru familiarizarea cu interfețe diferite. Se verifică în prealabil condițiile de acces, protecția datelor și cerințele privind conturile elevilor.
Instrument pentru vizualizarea execuției pas cu pas.	Urmărirea modificărilor valorilor variabilelor și a fluxului de execuție; înțelegerea algoritmului și sprijinirea procesului de depanare.	Poate fi utilizat inițial în activități ghidate și, ulterior, individual. Completează tabelul de urmărire a valorilor variabilelor și facilitează trecerea de la reprezentarea statică la observarea dinamică a execuției programului. Este suportul vizual al tabelului din E.0.2.
Vizualizări, animații și simulări ale algoritmilor	Observarea și compararea modului de funcționare a unor algoritmi, inclusiv a metodelor de căutare, de sortare sau a altor prelucrări prevăzute de programă.	Sunt utilizate pentru a susține înțelegerea și analiza algoritmului, nu pentru a înlocui construirea, urmărirea și explicarea acestuia de către elev. Pot constitui un punct de plecare pentru formularea de predicții și pentru compararea strategiilor. Recomandate explicit de programă;
Platformă pentru rularea și evaluarea automată a soluțiilor	Verificarea programelor pe mai multe seturi de date și furnizarea rapidă a informațiilor despre comportamentul soluției.	Este valorificată după ce elevul înțelege rolul testării și este capabil să propună propriile cazuri de testare. Rezultatul furnizat automat reprezintă o dovadă pentru revizuirea soluției; nu substituie analiza erorii și justificarea modificărilor efectuate.
Document sau spațiu digital colaborativ	Colectarea și organizarea exemplelor de erori, a strategiilor de depanare, a observațiilor și a soluțiilor identificate de elevi.	Poate deveni, progresiv, o resursă comună de învățare a clasei, construită din experiențele elevilor și reutilizată în activitățile ulterioare. Contribuțiile trebuie organizate astfel încât să poată fi identificată și valorificată participarea fiecărui elev.

E.1. „de la enunț la program”

ELEMENT	CONȚINUT
Competențe specifice vizate	CS 1.2 și CS 1.4 (identificarea etapelor și a elementelor de limbaj), cu intrare în CS 3.4 (utilizare)
Conținut din programă	2.1. Principii de elaborare a unui program — caracteristici ale gândirii computaționale; etapele analiză → proiectare → implementare → testare și depanare
Enunțul pentru elev	„Se citesc notele obținute de *n* elevi la un concurs. Afișați câți elevi au obținut peste 8 și media notelor acestora.”
Restricții	$1 \leq n \leq 30$; notele sunt numere naturale din intervalul [1, 10]. Dacă niciun elev nu are media peste 8, se afișează mesajul „niciun elev peste 8”, nu o valoare numerică.
Date de test (se dau elevilor)	(1) n = 5; notele 9 7 10 8 9 → se așteaptă **3** și **9.33** (2) n = 3; notele 5 6 7 → se așteaptă **mesajul**, nu 0.00 (3) n = 1; nota 10 → se așteaptă **1** și **10.00** (4) n = 4; notele 9 9 9 9 → se așteaptă **4** și **9.00**
Pașii orei (50 de minute)	— sarcinile de diagnostic D1–D3 (anexa D), fără anunț de test. — completarea frontală a fișei de analiză (E.0.1) pentru problema zilei; nicio linie de cod până când fișa este completă. — implementare individuală, cu profesorul în proximitate; mediul este deja instalat și testat. — se distribuie elevului pagina (anexa C) și se construiește, cu clasa, lista celor 5 pași de deblocare.
Produsul predat	Fișa de analiză completată + programul care rulează corect pe cele patru seturi de test + rezultatele sarcinilor D1–D3
Criteriile anunțate	**De bază:** completează fișa de analiză cu sprijin; programul rulează pe setul (1). **Consolidat:** completează fișa independent; programul rulează pe toate cele 4 seturi. **Avansat:** tratează cazul (2) fără să i se fi indicat problema împărțirii la zero și explică de ce apare.
Ce face profesorul	Nu corectează codul în timpul scrierii; notează în fișa D.3 distribuția pe cele trei niveluri. Anunță că aceleași criterii revin în testul din săptămâna 6.
Ce evită	Instalarea mediului în timpul orei; prezentarea programei; recapitularea sintaxei din gimnaziu.

E.2. „Același algoritm, trei reprezentări”

ELEMENT	CONȚINUT
Competențe specifice vizate	CS 2.2 (explicarea etapelor), CS 3.4 (utilizarea elementelor de limbaj)
Conținut din programă	2.1 — caracteristici de bază, avantaje și limite ale principalelor moduri de reprezentare a algoritmilor: blocuri grafice, pseudocod, limbaj de programare (cu interpretor sau compilator)
Enunțul pentru elev	„Pentru fiecare dintre cele trei probleme de mai jos, completați coloana lipsă din tabelul comparativ, apoi scrieți și rulați în limbajele Python și C++”
Cele trei probleme	(a) Se citesc două numere. Afișați-l pe cel mai mare. *Caz obligatoriu de tratat: numerele egale.* (b) Se citesc n numere. Afișați câte sunt pare. *Caz obligatoriu: n = 0.* (c) Se citesc n numere. Afișați valoarea minimă. *Caz obligatoriu: toate valorile sunt egale.*
Date de test	(a) (7, 7) → 7 · (–3, 2) → 2 · (5, 4) → 5 (b) 6; 1 2 3 4 5 6 → 3 · 0; (niciun număr) → 0 · 3; 2 4 6 → 3 (c) 4; 5 5 5 5 → 5 · 3; –1 0 1 → –1 · 1; 8 → 8
Pașii orei	— se proiectează același algoritm în cele trei reprezentări; elevii spun ce se pierde și ce se câștigă la fiecare trecere. — completarea tabelului comparativ (E.0.4), în perechi. — implementarea celor trei programe, cu rularea pe toate seturile de test. — se anunță formal criteriile pentru cele trei niveluri ale întregii unități 1 și se afișează în laborator.
Produsul predat	Tabelul comparativ completat individual + trei programe care rulează corect pe toate seturile de test
Criteriile anunțate	**De bază:** transferă algoritmul din pseudocod în Python și C++, folosind modelul de mai jos. **Consolidat:** completează independent toate cele trei reprezentări și tratează cazurile obligatorii. **Avansat:** explică de ce indentarea din Python și acoladele {} în C++ au valoare sintactică, spre deosebire de gruparea vizuală din blocurile grafice.
Ce face profesorul	Anticipează și explică explicit dificultățile de transfer specifice celor două limbaje — Python și C++ — înainte ca acestea să genereze erori de implementare: în Python, indentarea, conversia datelor citite prin input() și limitele intervalului generat de range(); în C++, declararea și tipizarea variabilelor, delimitarea blocurilor prin acolade, utilizarea separatorului; și formularea explicită a inițializării, a condiției și a actualizării în instrucțiunea for.

E.3. „Cifrele unui număr”

ELEMENT	CONȚINUT
Competențe specifice vizate	CS 1.2, CS 3.2 (algoritmi specializați pentru prelucrarea numerelor)
Conținut din programă	2.2. Prelucrări ale datelor numerice — operații specifice cu cifrele unui număr: acces la o cifră, adăugare a unei cifre la stânga, adăugare a unei cifre la dreapta, parcurgerea cifrelor; aplicarea algoritmilor de bază
Enunțul pentru elev	„Se citește un număr natural *n*. Scrieți două programe: (A) afișează suma cifrelor lui *n* și câte cifre pare are; (B) afișează oglinditul lui *n* (numărul cu cifrele în ordine inversă). Construiți singuri setul de teste și completați fișa de analiză, fără model.”
Restricții	$0 \le n \le 2\,000\,000\,000$. Pentru (B), zerourile de la finalul numărului dispar în oglinditul numărului (1200 → 21) — un comportament corect, de consemnat în analiză.
Date de test (minimul cerut)	(A) n = 2 486 → suma 20, cifre pare 4 n = 0 → suma 0, cifre pare 1 n = 7 → suma 7, cifre pare 0 (B) n = 1 234 → 4 321 n = 1 200 → 21 n = 5 → 5 n = 0 → 0 Caz-limită obligatoriu în ambele: n = 0 — ciclul „cât timp n > 0” nu se execută niciodată.
Pașii orei	**10 min** — se reia mecanismul: n % 10 dă ultima cifră, n // 10 elimină ultima cifră, iar r = r * 10 + c adaugă o cifră la dreapta. **30 min** — cele două programe, individual, cu fișa de analiză completată autonom. **10 min** — verificare încrucișată în perechi, pe setul de teste al colegului.
Produsul predat	Două programe funcționale, cu set propriu de teste + fișa de analiză completată fără ghidaj pas cu pas
Criteriile anunțate	**De bază:** implementează (A) după mecanismul reluat în oră. **Consolidat:** implementează ambele programe și tratează cazul n = 0. **Avansat:** explică de ce oglinditul lui 1200 este 21 și propune o variantă care păstrează zerourile, dacă cerința ar fi alta.
Ce face profesorul	Nu mai distribuie fișa de analiză completată parțial: aceasta este ora în care se verifică transferul ciclului complet analiză → testare pe o clasă nouă de probleme.

E.4. „Divizibilitate. Teste care pot infirma soluția”

ELEMENT	CONȚINUT
Competențe specifice vizate	CS 2.2, CS 3.2, cu intrare în CS 5.2 (evaluarea corectitudinii)
Conținut din programă	2.1 — 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
Enunțul pentru elev	„Se citește un număr natural *n*. Afișați toți divizorii lui *n* și verificați dacă numărul este perfect. **Construiți setul de teste înainte de a scrie programul.**”
Restricții	$1 \leq n \leq 10\,000$.
Setul de teste, pe trei categorii (se completează de elev, înainte de implementare)	**Teste normale:** n = 28 → divizorii 1 2 4 7 14 28; perfect. n = 12 → 1 2 3 4 6 12; nu este perfect. **Teste-limită:** n = 1 → divizorul 1; nu este perfect n = 6 → cel mai mic număr perfect. n = 10 000 → verifică timpul de execuție. **Test de eroare / date invalide: n = 0 sau** n negativ → programul trebuie să respingă datele, nu să afișeze un rezultat greșit.
Pașii orei	**10 min** — construirea setului de teste, în perechi, pe fișă, fără calculator. **20 min** — implementarea individuală și rularea pe propriul set de teste. **15 min** — depanare cu vizualizarea execuției pas cu pas (Python Tutor) și completarea jurnalului de depanare (E.0.3). **5 min** — se colectează erorile frecvente în jurnalul clasei.
Produsul predat	Setul de teste (normale, limită, de eroare), construit înainte de rulare, + programul + jurnalul de depanare cu cel puțin o eroare identificată și corectată de elev
Criteriile anunțate	**De bază:** construiește teste care confirmă soluția. **Consolidat:** construiește și teste-limită, și teste cu date invalide. **Avansat:** propune un test care ar putea infirma soluția și explică ipoteza pe care o verifică testul.
Ce face profesorul	Nu distribui setul de teste înainte ca elevi să îl construiască. Evaluează jurnalul de depanare, nu absența erorilor.

E.5. „Divizibilitate. Două soluții, aceeași problemă”

ELEMENT	CONȚINUT
Competențe specifice vizate	CS 4.2 și CS 4.3 (analiza comportamentului și a aplicabilității), cu intrare în CS 5.2
Conținut din programă	2.1 — eficiența algoritmilor: caracteristici din punctul de vedere al spațiului de memorie și al timpului de executare; criterii pentru determinarea ordinului de complexitate al unui algoritm polinomial (notația O)
Enunțul pentru elev	„Se citește un număr natural n. Afișați numărul divizorilor lui n, prin două soluții: (A) parcurgând toate valorile de la 1 la n; (B) parcurgând numai valorile până la $\sqrt{n}$ și numărând divizorii în perechi. Numărați operațiile efectuate de fiecare și comparați.”
Restricții	$1 \leq n \leq 1\,000\,000$. Se cere afișarea, pentru fiecare soluție, a numărului de iterații efectuate.
Tabelul de comparat (se completează de elev)	**Tabel comparativ al eficienței soluțiilor – de completat de elev** Elevul rulează cele două soluții pentru valori diferite ale lui n, consemnează **numărul de iterații** at **timpul de execuție**, apoi compară rezultatele obținute. n | Număr de iterații – soluția A | Număr de iterații – soluția B | Timp de execuție – A (ms) | Timp de execuție – B (ms) 100 | … | … | … | … 10 000 | … | … | … | … 1 000 000 | … | … | … | … Atenție la cazurile particulare: pentru valorile lui n care sunt pătrate perfecte (de exemplu, $n = 36$ sau $n = 10\,000$), divizorul $\sqrt{n}$ se ia în considerare **o singură dată**.
Pașii orei	**10 min** — se rulează ambele soluții pentru n = 100; diferența este nesemnificativă. Întrebarea orei: „Atunci de ce ne-ar interesa?” **20 min** — implementarea contorului de operații și completarea tabelului cu cele 3 valori. **15 min** — formularea concluziei: A este de ordinul O(n), B este de ordinul O(√n); pe date mici diferența nu se vede. **5 min** — biletul de ieșire (anexa F.3).
Produsul predat	Tabelul comparativ completat (număr de operații și timp măsurat) + concluzia scrisă, cu estimarea ordinului de complexitate pentru fiecare soluție
Criteriile anunțate	**De bază:** rulează ambele soluții și constată diferența de timp. **Consolidat:** leagă numărul de operații de dimensiunea datelor și numește ordinul. **Avansat:** explică de ce, pe date mici, soluția „mai slabă” poate fi preferabilă și tratează corect cazul pătratelor perfecte.
Ce face profesorul	Menține discuția la nivel intuitiv, prin numărarea operațiilor — programa nu cere formalizare matematică excesivă.

E.6. testul de autoevaluare și ora din rezervă

ELEMENT	CONȚINUT
Ora 1 — evaluare sumativă a unității 1	Testul din anexa F.2, aplicat integral. Se corectează în rubrica din F.1, pe cele trei niveluri, nu pe punctajul brut.
Ora 2 — activitate diferențiată / posibilă valorificare a rezervei de 25%.	Pe baza rezultatelor evaluării și a celorlalte dovezi de învățare colectate în unitate, profesorul decide dacă este necesară valorificarea unei ore din rezerva de 25%. În acest caz, selectează traseul adecvat: remediere (E.8), consolidare (E.9), aprofundare sau extindere (E.10).
Regula de decizie, pe scurt	Peste o treime sub „De bază” la criteriul „analiza problemei” sau „testarea” → E.8, cu profesorul la stația de remediere. Distribuție echilibrată, majoritatea la „Consolidat” → E.9. Peste un sfert la „Avansat” la cel puțin două criterii → E.10, în paralel cu E.8 pentru restul clasei (trei stații, cu rotație).
Produsul orei	Pentru fiecare elev: un produs pe criteriul lui slab + grila de autoevaluare (anexa F.4) completată + decizia scrisă a profesorului privind unitatea 2
Ce face profesorul	Consemnează în planificare că ora provine din rezerva de 25% și notează traseul dominant. Nu introduce conținut nou în această oră.

E.7. Fișă de rezervă — REMEDIERE („un pas o dată”)

ELEMENT	CONȚINUT
Cui se adresează	Elevilor situați sub „De bază” la criteriul „analiza problemei” sau „testarea”
Enunțul pentru elev	„Se citesc *n* numere. Afișați valoarea minimă și de câte ori apare ea.”
Sprijinul acordat	Fișa de analiză este completată pe jumătate: câmpurile „date de intrare” și „date de ieșire” sunt deja completate; elevul completează doar „date de manevră” și „cazuri-limită”. Tabelul de evoluție a valorilor este prezentat ca model pentru primele două iterații. Setul de teste este dat: 5; 3 1 4 1 5 → minim 1, apare de 2 ori · 3; 7 7 7 → 7, de 3 ori · 1; 9 → 9, o dată. Profesorul rămâne la această stație pe toată durata orei.
Produsul predat	Fișa de analiză completă + tabelul de evoluție completat până la final + programul care rulează pe cele trei teste
Cum verific efectul	Elevul reia aceeași structură pentru o a doua problemă, **fără fișă completată parțial**: „Câte numere sunt mai mari ca primul număr din șir?”. Dacă o duce la capăt singur, remedierea a funcționat.

E.8. CONSOLIDARE („aceeași problemă, alte restricții”)

ELEMENT	CONȚINUT
Cui se adresează	Majorității clasei, situată la nivelul „Consolidat”
Enunțul pentru elev	„Reluați problema temperaturilor din săptămâna 3, cu două modificări: (1) lista poate fi vidă; (2) se cere, în plus, a doua valoare ca mărime, distinctă de maxim.”
De ce funcționează	Structura problemei este cunoscută, dar contextul se schimbă: transferul se produce pe baza restricțiilor, nu pe baza algoritmului. Cerința (2) impune o condiție suplimentară — punctul în care se vede dacă elevul a înțeles ce memorează și de ce.
Date de test	5, 12, 18, 15, 20, 10 → media 15.0, maxim 20, a doua valoare 18 3, 7, 7, 7 → media 7.0, maxim 7, Nu există a doua valoare distinctă 5 → media 5.0, maxim 5, nu există a doua valoare
Produsul predat	Programul modificat + fișa de analiză actualizată, cu noile cazuri-limită + o propoziție despre ce a trebuit schimbat față de varianta din săptămâna 3
Cum verific efectul	Justificarea alegerii (a doua parcurgere sau condiție suplimentară) apare spontan în analiză, fără cerință explicită.

E.9. EXTINDERE („două complexități”)

ELEMENT	CONȚINUT
Cui se adresează	Elevilor situați la „Avansat” la cel puțin două criterii
Enunțul pentru elev	„Se citește un număr natural n. Verificați dacă este prim, prin două soluții: (A) încercând toți divizorii de la 2 la $n - 1$; (B) încercând numai divizorii până la $\sqrt{n}$. Estimați ordinul de complexitate al fiecăreia și stabiliți de la ce valoare a lui n diferența devine vizibilă în timpul de execuție.”
Restricții	$2 \leq n \leq 2\,000\,000\,000$. Se cere afișarea numărului de iterații pentru fiecare soluție.
Cerințe suplimentare	Tratarea corectă a cazurilor $n = 2$ și $n = 3$. Un tabel cu cel puțin patru valori ale lui n, alese de elev, care să arate pragul de la care diferența se observă. O concluzie de trei rânduri: $O(n)$ față de $O(\sqrt{n})$, cu precizarea că estimarea se referă la cazul cel mai defavorabil — numerele prime.
Produsul predat	Cele două probleme scrise în limbajele Python și C++, tabelul cu numărul de iterații și timpii măsurați și concluzia scrisă, cu estimarea ordinului
Variantă de extindere, la alegere	Redactarea „ghidului de depanare al clasei”: cele mai frecvente zece erori din jurnalul comun, fiecare cu mesajul, cauza reală și verificarea care o confirmă. Produsul se prezintă clasei în săptămâna 7 și rămâne o resursă pentru tot anul.
Cum verific efectul	Produsul poate fi utilizat de un coleg fără explicații suplimentare.

Anexa F. Evaluare și feedback pentru început

Criteriile se anunță în săptămâna 2 și se aplică identic la testul din săptămâna 6. Formularea descriptorilor respectă structura standardelor naționale de evaluare: performanța se descrie prin gradul de sprijin necesar și prin noutatea contextului, nu prin cantitatea de conținut reprodus.

F.1. Rubrica unității 1 — „Principii de elaborare a unui program”

CRITERIUL	DE BAZĂ	CONSOLIDAT	AVANSAT
1. Analiza problemei	Completează fișa de analiză după model sau cu sprijin; identifică intrarea și ieșirea.	Completează fișa independent, inclusiv datele de manevră, și justifică ce se memorează.	Identifică singur cazurile-limită neindicate în enunț și consemnează-le înainte de implementare.
2. Implementarea în Python și C++	Programele rulează pe setul de date principal; sintaxa este corectată cu sprijin.	Programele rulează pe toate seturile de date de test, cu structura cerută și cu comentarii.	Alege motivat o structură (de exemplu, o singură parcurgere în locul a două) și explică alegerea.
3. Testarea	Construiește teste care confirmă soluția.	Construiește teste normale, teste-limită și teste cu date invalide înainte de rulare.	Propune un test care ar putea infirma soluția și numește ipoteza pe care o verifică.
4. Depanarea	Localizează eroarea cu sprijin, folosind tabelul de evoluție a valorilor date parțial.	Localizează eroarea independent și completează jurnalul de depanare.	Distinge tipul erorii (de sintaxă, de logică, de execuție) și explică de ce mesajul primit nu indică locul real.
5. Proiectarea modulară	Descompune problema în module după modelul dat.	Numește rolul fiecărui modul și indică datele care intră și cele care ies din el.	Propune o descompunere alternativă, justificată prin refolosire sau prin claritate.
6. Eficiența	Constată, prin rulare, că o soluție este mai rapidă.	Leagă numărul de operații de dimensiunea datelor și numește ordinul de complexitate.	Explică de ce, pe date mici, soluția cu ordin mai mare poate fi preferabilă.

Nivelul de performanță se raportează la fiecare criteriu de reușită și nu se atribuie elevului în mod global. Același elev poate demonstra un nivel avansat la un anumit criteriu și poate avea nevoie de sprijin la altul. De aceea, deciziile privind valorificarea rezervei de 25% se fundamentează pe analiza rezultatelor la nivelul fiecărui criteriu și pe dificultățile sau progresele identificate, nu exclusiv pe o medie generală a rezultatelor.

F.2. Test de autoevaluare

Se anunță ca test de autoevaluare. Fiecare item indică nivelul atins la un criteriu, nu punctajul.

ITEMUL	SARCINA	CRITERIUL VIZAT
I1	„Se citește un număr natural. Afișați cel mai mare cifră a sa precum și media tuturor cifrelor sale pare.” Completați fișa de analiză: date de intrare, date de ieșire, date de manevră, restricții și cel puțin două cazuri-limită.	Criteriul 1 — analiza problemei

ITEMUL	SARCINA	CRITERIUL VIZAT
I2	Implementați soluția analizată la I1 și rulați-o pentru numărul: 14352→ cea mai mare cifră 5, media cifrelor pare 3.	Criteriile 2 și 5 — implementarea și organizarea codului
I3	Construiți pentru programul de la I2 un set de trei teste: unul normal, unul-limită și unul cu date invalide. Pentru fiecare, scrieți rezultatul așteptat **înainte** de a rula.	Criteriul 3 — testarea
I4	Se dă un program scurt care ar trebui să afișeze cifra maximă dintr-un număr, dar rezultatul nu este corect. Completați tabelul de evoluție a valorilor și indicați prima linie cu eroare și cauza acesteia.	Criteriul 4 — depanarea
I5	Programul de la I2 parcurge cifrele numărului o singură dată. Se poate determina și numărul de cifre egale cu cifra maximă? Argumentați în trei rânduri şi în caz afirmativ descrieţi soluţia.	Criteriul 6 — eficiența

Baremul pe niveluri — se citește pe orizontală, nu se adună

ITEM	DE BAZĂ	CONSOLIDAT	AVANSAT
I1	Date de intrare și date de ieșire corecte	Date de intrare, date de ieșire și date de manevră corecte	Date de intrare, date de ieșire și date de manevră corecte + două cazuri-limită
I2	Programul furnizează rezultatul corect pentru datele date	Programul furnizează rezultatul corect pentru datele date și structură clară	Programul furnizează rezultatul corect pentru datele date și structură clară și justificarea cerută
I3	un test pentru corectitudinea datelor	un test pentru corectitudinea datelor + test-limită și test cu date invalide	un test pentru corectitudinea datelor + test-limită și test cu date invalide + rezultatul așteptat scris înainte de rulare
I4	indică linia cu sprijin	indică linia și cauza independent	indică linia și cauza independent + numește tipul erorii și explică de ce mesajul nu indică locul real
I5	răspunde „da” fără justificare	răspunde „da” cu justificare	răspunde „da” cu justificare şi descrie soluţia

F.3. Biletul de ieșire (instrument pentru elev)

ÎNTREBAREA	RĂSPUNSUL TĂU
Ce am reușit să fac astăzi, concret?	……
La ce m-am blocat și ce am făcut?	……
Ce criteriu cred că am atins și la ce nivel?	……
O întrebare pe care o am pentru ora următoare	……

Se aplică la finalul unor activităţi de învăţare. Profesorul citește biletele între ore și începe ora următoare cu răspunsul la două-trei întrebări, nominalizând întrebarea, nu elevul.

F.4. Grila de autoevaluare a elevului (finalul unității 1)

ÎMI PROPUN SĂ POT…	ÎNCĂ NU	CU SPRIJIN	SINGUR(Ă)	POT EXPLICA ALTUIA
să completez fișa de analiză pentru o problemă nouă	□	□	□	□
să spun ce date trebuie memorate și de ce	□	□	□	□
să scriu în Python un program cu decizie și repetiție	□	□	□	□
să scriu în C++ un program cu decizie și repetiție	□	□	□	□
să construiesc teste înainte de a rula programul	□	□	□	□
să găsesc singur o eroare cu tabelul de evoluție a valorilor	□	□	□	□
să descompun o problemă în module și să spun ce face fiecare	□	□	□	□
să compar două soluții și să spun care e mai eficientă și de ce	□	□	□	□

Pasul meu următor, până la finalul unității 2: ……

F.5. Două modele de feedback descriptiv însoțit de feedforward

1. UNDE EȘTI (evidență observabilă)	2. UNDE TREBUIE SĂ AJUNGI (criteriul)	3. CUM AJUNGI (pasul următor, concret)
Pe un set de teste. „Ai trei teste și toate confirmă soluția: 28, 12 și 6 sunt numere pe care programul le tratează corect. Nu ai niciun test cu n = 1 și niciunul cu date invalide.”	Criteriul 3, nivelul „Consolidat”: setul conține teste normale, teste-limită și teste cu date invalide, construite înainte de rulare.	„Înainte de ora următoare, adaugă două linii în fișă: ce ar trebui să afișeze programul pentru n = 1 și ce ar trebui să facă pentru n = 0. Scrie răspunsul așteptat *înainte* de a rula, apoi rulează și compară.”

REGULA CELOR DOUĂ CICLURI Orice produs care se adaugă intră în portofoliu primește feedback pe o versiune intermediară și se revizuiește obligatoriu înainte de predare. Feedbackul care ajunge la elev odată cu nota descrie un rezultat închis și nu mai are niciun efect asupra învățării. Nota se însoțește întotdeauna de pasul următor.

Anexa G. Harta deciziilor pentru cei 25%

Rezerva de 25% din timpul alocat disciplinei, prevăzută de programă, oferă profesorului flexibilitatea necesară pentru adaptarea demersului didactic la nevoile de învățare și la progresul elevilor, identificate pe baza dovezilor colectate pe parcurs. În regim intensiv, la un buget anual de aproximativ 136 de ore (4 ore/săptămână), această rezervă reprezintă aproximativ 34 de ore, care pot fi valorificate, în funcție de nevoile identificate, pentru remediere, consolidare, aprofundare sau extindere.

Timpul disponibil nu presupune adăugarea automată de conținuturi, ci permite **diferențierea învățării, creșterea gradului de autonomie, abordarea unor sarcini cu niveluri diferite de complexitate, compararea și optimizarea soluțiilor, precum și dezvoltarea unor aplicații sau proiecte**, în acord cu competențele vizate de programă.

Harta de mai jos corelează dovezile de învățare colectate, nevoile identificate, deciziile didactice posibile, resursele adecvate și modalitățile de verificare a progresului, astfel încât utilizarea rezervei de 25% să fie fundamentată pe date privind învățarea și progresul elevilor, nu pe impresii generale sau pe o repartizare rigidă din timp.

DACĂ OBSERV…	POT DECIDE…	RESURSĂ / MODALITATE DE LUCRU	CUM VERIFIC EFECTUL
Un număr semnificativ de elevi întâmpină dificultăți în analiza problemei: identificarea datelor, a restricțiilor, a rezultatului așteptat sau a strategiei de rezolvare.	**Remediere** focalizată pe analiza și structurarea problemei, cu sprijin treptat din partea profesorului.	fișă de analiză parțial completată, exemplu modelat și seturi de date oferite; sprijinul este redus progresiv.	Elevul rezolvă o problemă similară cu mai puțin sprijin sau independent, identificând corect datele, restricțiile și strategia de rezolvare.
Dovezile arată dificultăți recurente în construirea și utilizarea cazurilor de test.	**Remediere** focalizată pe proiectarea testelor și pe anticiparea rezultatelor.	fișă adaptată pentru construirea și compararea unor cazuri uzuale, cazuri-limită și situații particulare relevante pentru problemă.	Elevul formulează rezultatul așteptat înainte de rulare, justifică alegerea testelor și utilizează rezultatele obținute pentru a verifica sau a revizui soluția.
Majoritatea elevilor demonstrează achizițiile vizate în contexte familiare, dar au nevoie de contexte variate pentru stabilizarea și transferul acestora.	**Consolidare**, prin modificarea datelor, a restricțiilor sau a contextului problemei.	problemă construită pe aceeași structură algoritmică, dar cu restricții modificate, situații particulare sau o cerință suplimentară.	Elevul transferă strategia într-o situație modificată și explică sau justifică adaptările realizate.
Un grup de elevi demonstrează constant autonomie și poate analiza, testa și argumenta soluțiile în contexte noi.	**Aprofundare sau extindere** prin sarcini care solicită compararea, optimizarea, argumentarea sau elaborarea unor soluții alternative.	compararea a două soluții, investigarea eficienței acestora, construirea unor cazuri de test relevante sau realizarea unei resurse de depanare reutilizabile pentru colegi.	Elevul compară și argumentează soluțiile pe baza unor criterii explicite și produce un rezultat care poate fi verificat și utilizat într-un context nou.
La același elev apar niveluri diferite de performanță pentru criterii diferite.	**Diferențiere în cadrul aceleiași activități**, în funcție de nevoia de învățare identificată, fără atribuirea unui nivel global elevului.	Sarcini sau stații de lucru cu niveluri diferite de sprijin și complexitate, valorificând selectiv resurse deja elaborate. Profesorul acordă sprijin prioritar acolo unde dificultatea este mai mare.	Se colectează o nouă dovadă pentru criteriul care a determinat intervenția și se compară cu performanța anterioară pentru a observa progresul.

DACĂ OBSERV…	POT DECIDE…	RESURSĂ / MODALITATE DE LUCRU	CUM VERIFIC EFECTUL
Programele sunt funcționale, dar dovezile privind testarea, explicarea sau documentarea soluției sunt insuficiente.	**Clarificarea și recalibrarea criteriilor de reușită,** fără reluarea inutilă a conținuturilor algoritmice deja demonstrate.	Activitate scurtă de analiză a unui exemplu și a unui contraexemplu pentru a clarifica ce dovezi trebuie să însoțească soluția: funcționarea, testarea, explicarea și, după caz, documentarea acesteia.	În activitatea următoare, elevul furnizează toate dovezile solicitate de criteriile comunicate, fără sprijin suplimentar pentru structurarea produsului.