REPERE METODOLOGICE

pentru aplicarea noului curriculum liceal

INFORMATICĂ

Clasa a IX-a • Curriculum de specialitate • 2026–2027

De la algoritm la produs software: continuitate profesională, limbaj nou de programare, instrumente noi.

IDEEA CENTRALĂ	Trecerea la limbajul Python este o schimbare vizibilă, dar nu anulează experiența profesorului format în alte limbaje de programare. Analiza problemei, gândirea algoritmică, controlul execuțării, testarea, depanarea, eficiența și modularizarea rămân nucleul profesional. Ghidul sprijină transferul către utilizarea noului limbaj, a modelelor explicite de organizare a datelor și a unor abordări pedagogice care încurajează explorarea, formularea de ipoteze, experimentarea și descoperirea, fără a deveni un curs de Python.

CUM SE CITEȘTE RAPID

ANCORE	SCHIMBĂRI	TRANSFER	APLICARE	REFLECȚIE
proiectarea algoritmilor, elemente specifice limbajelor de programare	modele explicite de organizare a datelor, Python devine limbaj de programare obligatoriu	conținuturi studiate la alte clase; alte limbaje de programare → Python	laborator, teste, proiecte, IA	Ce păstrez? Ce verific? Ce revizuiesc?

01 Cum citim cele trei programe?	02 Ce se schimbă, de fapt?
03 Ce rămâne și poate fi valorificat?	04 Trei specializări, un nucleu comun
05 Cu ce vin elevii din gimnaziu?	06 Cum folosim standardele definite pentru clasa a VIII-a?
07 Cum citim algoritmii specializați pe clase de probleme din programă?	08 Diagnoza limbajului de programare fără a confunda limbajul cu proiectarea algoritmilor
09 Programul ca proces: analiză–proiectare–implementare–testare–depanare	10 Predicția înainte de rulare
11 Depanarea ca metodă de învățare	12 Atelierul de teste: încercăm să găsim punctele slabe ale programului
13 De la alte limbaje de programare utilizate la gimnaziu, la Python: transfer, nu restart	13.1 Primele ore ale profesorului cu Python
13.2 Armonizare cu Python a limbajelor de programare utilizate la gimnaziu: Intrare, ieșire, tipuri și conversii	13.3 Armonizare cu Python a limbajelor de programare utilizate la gimnaziu: Structuri alternative și operatori logici
13.4 Armonizare cu Python a limbajelor de programare utilizate la gimnaziu: Repetiția cu for, range, while	14 Prelucrări ale datelor numerice
15 Proiectarea modulară	16 Subprograme - de la alte limbaje de programare la Python
17. Introducere în programarea orientate pe obiecte	18 Interfața grafică și Tkinter în Python
19 Fișiere text - de la alte limbaje de programare la Python	20 Modelul conceptual liniar

20.1 Listele în Python: zona în care diferențele devin importante	20.2 Mutabilitate, copiere și efecte colaterale
20.3 Generarea sistematică a elementelor unei liste	20.4 Metodele de sortare: algoritmi, comportament, eficiență
21 Eficiența și notația O	22 Dincolo de „explic codul – scrieți și voi”
23 Programarea în perechi și revizuirea de cod	24 IA ca ajutor de tranziție profesională
25 Cum cerem IA să ne ajute să învățăm Python?	26 IA pentru generarea materialelor didactice
27 Cum verificăm o resursă generată cu IA?	28 IA cu elevii: ce poate face și ce nu delegăm
29 Alte resurse pentru adaptarea profesorului la noul curriculum	30 Rezerva curriculară de 25%
31 Cum construim parcursul anual la specializarea matematică-informatică?	32 Cum construim parcursul anual la specializarea științe ale naturii?
33 Cum construim parcursul anual la specializarea matematică-informatică militară?	34 Interdisciplinaritate reală
35 Evaluarea care produce învățare	36 Feedback care lasă elevului efortul cognitiv
37 Diferențiere fără trei programe paralele	38 De la program la clasă: două microsecvențe
39 Ce mai putem folosi din Reperele metodologice elaborate în anul 2021?	40 Întrebări pentru proiectarea didactică
41 Documente de referință și anexa Excel	42 Notă privind realizarea documentului

01 CUM CITIM CELE TREI PROGRAME?

Un ghid comun, trei trasee de curriculum de specialitate.

Cele trei programe pentru clasa a IX-a au aceeași arhitectură bazată pe câte șase competențe generale și câte 30 de competențe specifice. Diferențele apar mai ales în alocarea orară, unele exemple de activități de învățare și unele conținuturi/grade de aprofundare. Disciplina Informatică este definită ca o continuare a disciplinei Informatică și TIC din gimnaziu, cu progres spre modele de organizare a datelor, algoritmi specializați pe clase de probleme, proiectare modulară, eficiență și produse software, precum și elemente de limbaj de programare pentru implementarea algoritmilor care utilizează modelele studiate de organizare a datelor.

PROFIL	ORE/SĂPT.	PARTICULARITATE
Matematică-informatică	2 h: 1 h pentru studiu teoretic și 1 h pentru activități practice	Prelucrări ale datelor numerice, organizare a datelor în liste (inclusiv cazuri particulare ca stivă/coadă), subprograme, metode de sortare, fișiere text, Tkinter și traseu mai amplu
Științe ale naturii	1 h pentru studiu teoretic și pentru activități practice	Prelucrări ale datelor numerice, organizare a datelor în liste, subprograme, metode de sortare de bază, fișiere text și nucleu mai concentrat, cu prioritate pe aplicare și contexte de științe
Matematică-informatică militară	3 h: 1 h pentru studiu teoretic și 2 h pentru activități practice	Prelucrări ale datelor numerice (inclusiv ciurul lui Eratostene, exponențiere rapidă), organizare a datelor în liste, subprograme, metode de sortare, fișiere text, Tkinter și aprofundări suplimentare

DE REȚINUT	Activitățile practice sunt prevăzute să se desfășoare obligatoriu în laboratorul de informatică. Toate cele trei programe au o rezervă de 25% din timpul alocat disciplinei pentru remediere, consolidare, aprofundare sau extindere.

02 CE SE SCHIMBĂ, DE FAPT?

Nu doar limbajul. Se schimbă și nivelul de explicitare a procesului software.

2009	2025	SCHIMBARE DE ACCENT
C++ era limbajul dominant utilizat în practică	Python este limbaj de programare obligatoriu	mai puține detalii de sintaxă, mai mult spațiu pentru gândire
algoritmi și implementare	modele de organizare a datelor + algoritmi + modularizare + eficiență + testare	procesul de dezvoltare devine mai vizibil
pseudocod și programare structurată	subprograme, fișiere text, clase predefinite pentru liste în Python; Tkinter pentru unele specializări	mai multă apropiere de practica actuală
evaluarea soluției corecte	corectitudine + eficiență + aplicabilitate + creare	progres cognitiv mai explicit
aplicații interdisciplinare	contexte STEM, economie, științe sociale, arte digitale	interdisciplinaritate mai largă

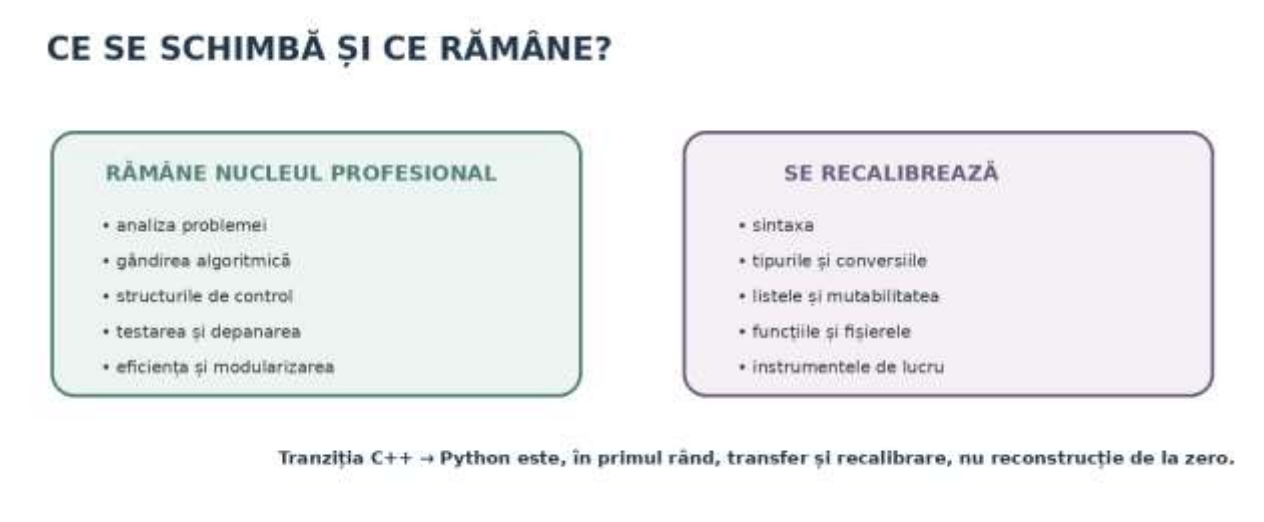
IMPORTANT

Schimbarea de limbaj este cea mai vizibilă, dar nu este singura. Programa urmărește explicit trecerea de la identificare și explicare la utilizare, analiză, evaluare și elaborare de soluții.

03

CE RĂMÂNE ŞI POATE FI VALORIFICAT?

Experienţa cu C/C++ este capital profesional transferabil.



RĂMÂNE VALABIL	Competenţele de specialitate fundamentale ale unui cadru didactic — analiza problemelor, proiectarea algoritmilor, testarea şi depanarea — reprezintă nucleul dificil al informaticii. Python cere o recalibrare a exprimării, nu abandonarea logicii profesionale.

04 TREI SPECIALIZĂRI, UN NUCLEU COMUN

Evităm și uniformizarea, și trei ghiduri complet separate.

COMUN	MATE-INFO	ȘTIINȚE ALE NATURII	MILITAR
6 CG / 30 CS	2 h	1 h	3 h
Prelucrări ale datelor numerice, organizare a datelor în liste, subprograme, metode de sortare (selecția minimului, liste de frecvențe), fișiere text, limbajul de programare Python	Cazuri particulare ale listelor (stivă/coadă), metode de sortare (metoda bulelor), Tkinter	nucleu concentrat	Cazuri particulare ale listelor (stivă/coadă), prelucrări particulare ale datelor numerice (Ciurul lui Eratostene, exponențiere rapidă), metode de sortare (metoda bulelor), Tkinter
testare și depanare	mai mult timp de aprofundare	selecție foarte atentă	mai multe aplicații și contexte specifice

DECIZIE DE PROIECTARE	Folosim aceeași logică metodologică pentru toate specializările, dar nu transferăm automat un conținut dintr-o programă în alta. Anexa Excel păstrează separat CG–CS–EAI și conținuturile integrale pentru fiecare programă.

05

CU CE VIN ELEVII DIN GIMNAZIU?

Pentru prima dată, nu mai pornim de la premisa că informatica începe în clasa a IX-a.

Programa de gimnaziu alocă aproximativ jumătate din timpul disciplinei formării competențelor algoritmice și de programare. În clasa a VIII-a, elevii identifică șiruri de numere, utilizează algoritmi de bază pentru prelucrarea acestora în construirea soluțiilor și îi implementează într-un mediu de programare. Limbajul de programare utilizat putea diferi de la o clasă la alta; ceea ce trebuie observat la intrare este controlul algoritmic, nu fidelitatea față de o sintaxă anume.

ANCORĂ	SARCINĂ SCURTĂ	CE OBSERV
șiruri de numere	identifică datele și regula într-un context	modelare și parcurgere
algoritm de bază	scrie pașii pentru o problemă simplă cu algoritmi de bază	elemente specifice ale algoritmilor de bază (inițializare, actualizare a rezultatului)
implementare	scrie cod în limbajul cunoscut	separarea algoritmului de sintaxa limbajului
testare	propune trei seturi de date de test	caz normal și limite
proiect	explică etapele unui mini-produs	autonomie și planificare

06 CUM FOLOSIM STANDARDELE DEFINITE PENTRU CLASA A VIII-A?

Ca hartă de autonomie și complexitate, nu ca test gata făcut.

NIVEL	LECTURĂ PEDAGOGICĂ	ADAPTARE
De bază	lucrează cu sprijin și pași ghidați	ofer șablon, exemple, conținuturi simplificate
Consolidat	lucrează independent în contexte familiare	retrag sprijinul, cer explicație și testare
Avansat	transferă și lucrează autonom în contexte noi	expun problema, cer comparație și optimizare

ANCORĂ DIN GIMNAZIU	
	Standardele definite pentru clasa a VIII-a includ explicit analiza șirurilor de numere, utilizarea algoritmilor de bază în construirea soluțiilor și implementarea acestora. Ele ne ajută să calibrăm sprijinul, nu să presupunem că toți elevii au studiat același limbaj.

07 CUM CITIM ALGORITMII SPECIALIZAȚI PE CLASE DE PROBLEME DIN PROGRAMĂ?

De la „știu codul” la „înțeleg clasa de probleme, comportamentul și alegerea”.

NIVEL DE PROGRES	ÎNTREBARE
Identifică	Ce problemă rezolvă și care sunt etapele?
Explică	De ce funcționează etapele?
Utilizează	Pot aplica algoritmul în context?
Analizează	Ce avantaje, limite și comportament are?
Evaluează	Este corect și eficient pentru situația dată?
Elaborează	Pot construi/adapta o soluție?

08

DIAGNOZA LIMBAJULUI DE PROGRAMARE FĂRĂ A CONFUNDA LIMBAJUL CU PROIECTAREA ALGORITMILOR

Întrebarea utilă este „Ce poate face elevul?", nu „În ce limbaj a lucrat?".

SARCINĂ	PERMITEM	INTERPRETĂM
maximul dintr-un șir dat cu număr mic de elemente	pseudocod, Python, C/C++, Scratch etc.	dacă algoritmul e corect, sintaxa se armonizează separat
urmărire a execuării algoritmului	orice reprezentare clară	model mental și controlul variabilelor
cazuri de test	limbaj natural	înțelegerea specificației
o eroare simplă	explicare orală/scrisă	depanare și cauzalitate

CE EVITĂM	
	Un test inițial care penalizează masiv un elev pentru că nu știe încă sintaxa Python, deși poate construi și explica algoritmul.

09 PROGRAMUL CA PROCES

Analiză → proiectare → implementare → testare → depanare devine o rutină profesională.

ETAPĂ	ÎNTREBAREA ELEVULUI
Analiză	Ce intră, ce trebuie să obțin, ce restricții am?
Proiectare	Care sunt pașii și ce structură de date aleg?
Implementare	Cum exprim în Python soluția proiectată?
Testare	Ce cazuri pot confirma sau infirma corectitudinea?
Depanare	Unde apare prima abatere și de ce?
Revizuire	Ce schimb pentru claritate sau eficiență?

10 PREDICȚIA ÎNAINTE DE RULARE

Rularea imediată poate ascunde lipsa modelului mental.

ÎNAINTE DE RUN	DUPĂ RUN
prezic ieșirea	compar predicția cu ieșirea
completez 3 pași într-un tabel de trasare	identific primul loc unde m-am înșelat
spun ce se întâmplă cu lista	observ efectul real asupra listei
estimez câte iterații sunt	verific prin contorizare

11	**DEPANAREA CA METODĂ DE ÎNVĂȚARE** Eroarea este o fereastră spre modelul mental al elevului.

TIP	EXEMPLU DE ÎNTREBARE
sintaxă	Ce regulă de scriere este încălcată?
logică	Care este primul pas în care valoarea devine greșită?
executare	Ce date produc eroarea și de ce?
specificație	Programul rezolvă problema cerută sau doar una asemănătoare?

FEEDBACK	„Nu rescrie programul. Urmărește valorile variabilelor până la prima abatere. Corectează doar acel pas și rulează din nou aceleași teste.”

12 ATELIERUL DE TESTE: ÎNCERCĂM SĂ GĂSIM PUNCTELE SLABE ALE PROGRAMULUI

Testarea este o competență, nu o formalitate de după cod.

TIP DE TEST	EXEMPLU
caz obișnuit	date care reflectă situația tipică
limită	0, 1, lungime minimă/maximă relevantă
caz advers	date care pun la încercare o presupunere
regresie	test care a descoperit o eroare și rămâne în set după corecție

13

DE LA ALTE LIMBAJE DE PROGRAMARE UTILIZATE LA GIMNAZIU, LA PYTHON: TRANSFER, NU RESTART

Punem în paralel concepte, nu traducem mecanic linii de cod.

ORE	FOCALIZARE	PRODUS
1–3	Gândire computațională, algoritmi - în paralel, reprezentare în pseudocod, schema logică, implementare în limbajul de programare cunoscut și în Python cu accent pe citire, scriere, conversii, operatori relaționali, operatori logici, atribuiri, instructiunea alternativă	3-6 programe
1 - 3	Prelucrări cu algoritmi de bază a unor șiruri de numere date – în paralel, reprezentare în pseudocod, schema logică, implementare în limbajul de programare cunoscut și în Python cu accent pe instrucțiuni repetitive	3-6 programe

CHEIE	
	Pentru armonizarea cu Python a limbajelor de programare utilizate în gimnaziu, compararea acestora în primele săptămâni este utilă ca punte. Pe termen lung, obiectivul este ca profesorul și elevul să utilizeze direct modelul și convențiile limbajului Python, nu să traducă permanent.

13.1 PRIMELE ORE ALE PROFESORULUI CU PYTHON

Un traseu autonom, moderat și realist.

13.2 ARMONIZARE CU PYTHON A LIMBAJELOR DE PROGRAMARE UTILIZATE LA GIMNAZIU: INTRARE, IEȘIRE, TIPURI ȘI CONVERSII

Python simplifică declarațiile, dar face conversia datelor vizibilă.

Citirea unui întreg și a unui număr real

C++	Python
`int n; double x;` `cin >> n >> x;`	`n = int(input())` `x = float(input())`

CAPCANĂ	
	input() produce șir de caractere. Trebuie să se facă explicit conversia atunci când problema lucrează cu numere. Această diferență este o bună ocazie pentru a discuta despre reprezentarea datelor, nu doar despre sintaxă.

13.3	ARMONIZARE CU PYTHON A LIMBAJELOR DE PROGRAMARE UTILIZATE LA GIMNAZIU: STRUCTURA ALTERNATIVĂ ȘI OPERATORI LOGICI Conceptul rămâne; se schimbă notația și rolul indentării.

Condiție compusă

C++	Python
if (x>=0 && x<10) k++;	if x >= 0 and x < 10: k += 1

CHEIE DE VERIFICARE	Elevul/profesorul trebuie să poată spune mai întâi condiția în limbaj natural, apoi să o exprime corect în Python. Traducerea simbol cu simbol este doar o etapă de tranziție.

13.4 ARMONIZARE CU PYTHON A LIMBAJELOR DE PROGRAMARE UTILIZATE LA GIMNAZIU: REPETIȚIA CU FOR, RANGE, WHILE

Cea mai frecventă capcană: intervalul produs de range.

INTENȚIE	PYTHON	ÎNTREBARE DE CONTROL
0,1,…,n-1	range(n)	de ce nu apare n?
1,2,…,n	range(1,n+1)	ce înseamnă capătul exclus?
a,a+p,…	range(a,b,p)	ce condiție trebuie să îndeplinească pasul?
repetă cât timp condiția e adevărată	while condiție:	ce modificare garantează terminarea?

14 PRELUCRĂRI ALE DATELOR NUMERICE

Utilizarea algoritmilor de bază pentru prelucrarea datelor numerice

- parcurgerea cifrelor și construirea unui număr;
- divizori, multipli și descompunere în factori primi;
- algoritmul lui Euclid;
- transformări de bază la specializările unde sunt prevăzute;
- Ciurul lui Eratostene și exponențiere rapidă la specializarea matematică-informatică militară.

ORE	FOCALIZARE	PRODUS
3-4	Prelucrarea datelor numerice, cu accent pe operații specifice cu cifrele unui număr	algoritmi de prelucrare a cifrelor unui număr
4-5	Algoritmi de divizibilitate, cu accent pe operații specifice cu divizorii unui număr	algoritmi de divizibilitate

SE SCHIMBĂ ACCENTUL	Valorile care se prelucrează trebuie obținute, pe baza unor prelucrări, fără a fi necesară memorarea acestora ca o colecție.
RĂMÂNE VALABIL	Baza algoritmică din vechea programă și din experiența de concurs/probleme rămâne utilă. Noutatea este să facem mai vizibile alegerea, testarea și eficiența soluției.

15 PROIECTAREA MODULARĂ

Un program bun poate fi explicat și pe părți.

ÎNTREBARE CENTRALĂ	Ce responsabilitate are fiecare modul și ce date primește/produce?

SEMN DE MODULARIZARE SLABĂ	INTERVENȚIE
o funcție citește, calculează, afișează și modifică date globale	separă responsabilitățile
aceeași secvență apare de trei ori	extrage un subprogram
parametrii nu sunt clari	numește rolul datelor și rezultatelor
funcția depinde de multe variabile globale	reduce dependențele și transmite explicit datele

16	**SUBPROGRAME: DE LA ALTE LIMBAJE LA PYTHON** Modularizarea rămâne aceeași idee; convențiile limbajului sunt mai ușoare.

Funcţie cu rezultat

C++	Python

int cmmdc(int a,int b){
  while(b){ int r=a%b; a=b; b=r; }
  return a;
}
	def cmmdc(a, b):
	  while b:
	  a, b = b, a % b
	  return a

DECIZIE DIDACTICĂ	Nu este necesar ca prima versiune Python să folosească simultan toate facilitățile idiomatice. Putem porni de la o versiune apropiată de logica deja cunoscută și apoi trece la o versiune mai expresivă.

17 INTRODUCERE ÎN PROGRAMAREA ORIENTATĂ PE OBIECTE

CHEIE DE VERIFICARE	
	Programarea orientată pe obiecte se poate introduce prin analogii cu prelucrarea personajelor în Scratch, precum și cu analogii intuitive cu lumea reală, evidențiind faptul că orice obiect are proprietăți (date) și comportamente (metode). **Predarea are în vedere doar noțiuni strict necesare pentru utilizarea, la nivel elementar, a unor clase predefinite.**

18 INTERFAȚA GRAFICĂ ȘI TKINTER ÎN PYTHON

Relevantă pentru specializarea matematică-informatică și matematică-informatică militară; nu devine curs de design de interfețe.

OBIECT	ROL DIDACTIC
Tk	fereastră principală
Label	afișarea unui text
Entry / Text	introducerea datelor
Button	declanșarea unei acțiuni
Frame	organizarea elementelor
Canvas	suprafață pentru desen/afișare
pack / grid / place	organizarea în fereastră

LIMITĂ	
	Interfața trebuie să servească funcției informatice. Nu consumăm disproporționat timp pentru aspectul vizual dacă intenția curriculară este modelarea, programarea și testarea.

19	**FIȘIERE TEXT – DE LA ALTE LIMBAJE DE PROGRAMARE LA PYTHON** Un context bun pentru a lega datele de intrare/ieșire de persistență și testare.

Citire din fișier

C++	Python

ifstream fin("date.in");
int n; fin >> n;
fin.close();
	with open("date.in", "r") as f:
	  n = int(f.read())

CHEIE	Programa cere principii de lucru și utilizarea clasei TextIOWrapper. Pentru clasa a IX-a, miza este fluxul datelor și folosirea corectă a operațiilor de deschidere/citire/scriere/închidere, nu explorarea exhaustivă a API-ului.

20 MODELUL CONCEPTUAL LINIAR

Organizarea datelor – proiectarea algoritmului înainte de implementarea în limbaj de programare

ORE	FOCALIZARE	PRODUS
1-2	Organizarea datelor – șiruri de valori care intervin în rezolvare, cu accent pe cazurile în care acestea se pot prelucra pe măsura parcurgerii lor sau este necesară reluarea parcurgerii acestora	Modele de organizare a datelor

CHEIE DE VERIFICARE	
	Vizează reprezentări abstracte ale modului în care sunt structurate și relaționate datele, înainte ca ele să fie memorate efectiv printr-un program. Ele răspund la întrebarea: „Ce informații trebuie să prelucrez/stochez și cum sunt legate între ele?” și nu la întrebarea „Ce tip de variabile utilizez pentru a le stoca concret în memorie?”

20.1 LISTELE ÎN PYTHON: ZONA ÎN CARE DIFERENȚELE DEVIN IMPORTANTE

Lista Python este flexibilă, dinamică și mutabilă.

OPERAȚIE	PYTHON	CE DISCUTĂM
creare	a = []	listă vidă
adăugare	a.append(x)	modifică lista existentă direct (in-place) și nu returnează o listă nouă (returnează None)
acces	a[i]	indexare de la 0
apartenență	x in a	operator de apartenență
lungime	len(a)	funcție, nu metodă a clasei list
sortare	a.sort()	modifică lista
copie	a.copy()	listă distinctă, copiere superficială

SE SCHIMBĂ ACCENTUL	Programa folosește lista nu doar ca tehnică de memorare, ci ca model conceptual pentru organizarea datelor. Înțelegerea accesului, ordinii și mutabilității este mai importantă decât memorarea unei colecții de metode.

20.2 MUTABILITATE, COPIERE ŞI EFECTE COLATERALE

O diferență mică de sintaxă poate ascunde o diferență conceptuală importantă.

Copierea unei colecţii

C++

```
int a[2]={1,2}, b[2];
for(int i=0;i<=1;i++)
    b[i]=a[i];
b[0]=9;
//  a ramane {1,2}
//  b devine {9,2}
```

Python

```
a = [1,2]
b = a
b[0] = 9
# a devine [9,2]
# b devine [9,2]


a = [1,2]
b = a.copy()  # copie distinctă
b[0] = 9
# a rămâne [1,2]
# b devine [9,2]
```

PENTRU PROFESOR

Acesta este un exemplu foarte bun în care IA poate genera explicații sau exemple, dar profesorul trebuie să ruleze efectiv codul și să observe comportamentul. Nu este suficientă analogia cu tabloul unidimensional în limbajul C++.

20.3 GENERAREA SISTEMATICĂ A ELEMENTELOR UNEI LISTE

Utilizarea modelului liniar de organizare a datelor fără memorarea acestora ca o colecție

ORE	FOCALIZARE	PRODUS
3 - 5	Algoritmi de generare sistematică a elementelor unei liste, fără a fi necesară memorarea acestora ca o colecție	algoritmi de generare a elementelor unei liste

SARCINĂ	DOVADĂ
generează n valori după o regulă	cod + primele valori urmărite manual
explică dependența dintre termeni	diagramă / tabel de trasare
aplică într-un context	interpretarea rezultatului
compară două modalități	argument privind claritatea/eficiența

SE SCHIMBĂ ACCENTUL	Valorile care se prelucrează trebuie generate, pe baza unor prelucrări, fără a fi necesară memorarea acestora ca o colecție.

20.4 METODELE DE SORTARE: ALGORITM, COMPORTAMENT, EFICIENȚĂ

Nu memorăm corpul unei funcții de sortare fără a înțelege ce face fiecare parcurgere.

METODĂ	CE URMĂRIM
selecția minimului	elementul selectat la fiecare pas; numărul de comparații
metoda bulelor	compararea elementelor adiacente și interschimbarea valorilor
listă de frecvențe	condiția de aplicabilitate și relația valoare–frecvență

CHEIE	
	Profesorul poate folosi instrumente pentru vizualizarea algoritmilor sau poate cere IA să genereze o trasare. Dar, înainte de a utiliza rezultatul la clasă, verifică manual cel puțin un caz și se asigură că reprezentarea folosește aceeași convenție ca lecția.

21

EFICIENȚA ȘI NOTAȚIA O

În clasa a IX-a, elevul trebuie să poată compara și argumenta, nu să recite definiții abstracte.

SITUAȚIE	ÎNTREBARE
divizori până la n vs până la √n	Cum se schimbă numărul de pași?
două sortări	Care efectuează mai multe comparații/interschimbări?
memorare vs prelucrare directă	Am nevoie să păstrez toate datele?
subprograme	Claritatea și reutilizarea justifică descompunerea?

22	**DINCOLO DE „EXPLIC CODUL – SCRIEȚI ȘI VOI”** Explicația rămâne utilă, dar nu trebuie să monopolizeze proiectarea algoritmică.

PROFESORUL POATE...	ELEVUL TREBUIE SĂ...
modela 5 minute un exemplu	prezică, urmărească și explice execuția
arăta o regulă de sintaxă	aleagă unde și de ce o folosește
oferi un cod funcțional	construi teste și găsi limite
oferi două soluții	compara și argumenta alegerea
oferi feedback	revizui o versiune

PRINCIPIU	O activitate este „activă” atunci când gândirea computațională aparține elevului: el modelează, alege, prezice, testează, explică și revizuiește.

23 PROGRAMAREA ÎN PERECHI ȘI REVIZUIREA DE COD

Cooperarea are sens dacă distribuie gândirea, nu tastatura.

ROL	SARCINĂ
conducător	scrie codul și verbalizează intenția
verificator	urmărește cerința, starea și testele
schimb de rol	la 8–12 minute sau la finalul subproblemei
revizuire	perechea explică un punct bun și un risc al soluției

CHEIE DE VERIFICARE	La final, fiecare elev trebuie să poată explica soluția. Dacă unul „a programat”, iar celălalt doar a privit, cooperarea nu și-a atins scopul.

24	**IA CA AJUTOR DE TRANZIȚIE PROFESIONALĂ** Un sprijin eficient dacă rămâne sub controlul profesional al profesorului.

MESAJ	IA nu este un substitut pentru competența profesorului și nici o autoritate asupra corectitudinii codului. Este însă foarte potrivită pentru explicații comparate, variații de exemple, teste, diagnostic de erori și generarea de material de lucru care poate fi verificat rapid.

25 CUM CEREM IA SĂ NE AJUTE SĂ ÎNVĂȚĂM PYTHON?

Promptul bun limitează, structurează și cere verificabilitate.

SCOP	MODEL DE CERERE
traducere	„Transformă acest algoritm C++ în Python. Păstrează algoritmul și explică diferențele semantice și sintactice.”
contrast	„Dă o versiune apropiată de C++ și una mai naturală în Python. Compară avantajele didactice.”
depanare	„Nu repara direct. Indică primul pas în care starea devine greșită și propune un test care evidențiază eroarea.”
teste	„Propune caz obișnuit, limită și advers; precizează rezultatul așteptat.”
învățare	„Explică mutabilitatea listelor unui profesor care cunoaște vectorii în C++. Dă un contraexemplu minimal.”

REGULĂ	
	Nu folosesc ca material de clasă un răspuns IA pe care nu l-am rulat/verificat și pe care nu îl pot explica independent.

26

IA PENTRU GENERAREA MATERIALELOR DIDACTICE

De la economie de timp la material didactic validat.

POATE GENERA	PROFESORUL ADAUGĂ
exemple gradate	CS, nivel, timp și context
cod lacunar	unicitatea/acceptabilitatea soluțiilor
programe cu erori	erori autentice și traseu de diagnostic
seturi de teste	validarea rezultatelor așteptate
rubrici	criterii observabile și relevante curricular
feedback	specificitate, adevăr tehnic și spațiu pentru revizuire

27	**CUM VERIFICĂM O RESURSĂ GENERATĂ CU IA?** Fluența nu este criteriu de corectitudine.

- Rulează codul în mediul pe care îl vor folosi elevii.
- Verifică rezultatele pentru cel puțin un caz obișnuit și două cazuri-limită.
- Verifică dacă algoritmul rezolvă specificația, nu doar exemplele.
- Elimină facilități Python care depășesc inutil nivelul urmărit.
- Verifică formularea în raport cu CS și conținuturile obligatorii.
- Corectează terminologia nefirească sau traducerile stângace.
- Asigură-te că elevului îi rămâne muncă de gândire.
- Verifică sursele, datele personale și drepturile de autor.

PRAG PROFESIONAL	Dacă profesorul nu poate explica de ce soluția este corectă și adecvată, resursa nu este încă pregătită pentru clasă.

28

IA CU ELEVII: CE POATE FACE ȘI CE NU DELEGĂM

Competența se dezvoltă prin judecată asupra rezultatului, nu prin simpla obținere a lui.

UTILIZARE POTRIVITĂ	CONDIȚIE
generează o soluție alternativă	elevul o testează și o compară
propune cazuri de test	elevul verifică rezultatele
explică o eroare	elevul reproduce și confirmă diagnosticul
generează cod	elevul poate explica fiecare parte relevantă
sugerează refactorizare	elevul justifică acceptarea sau respingerea

PRINCIPIU IA	IA poate produce cod; elevul trebuie să producă înțelegere, testare și judecată.

29 ALTE RESURSE PENTRU ADAPTAREA PROFESORULUI LA NOUL CURRICULUM

IA este una dintre căi, nu singura.

RESURSĂ	UTILITATE
documentația Python	referință precisă pentru sintaxă și biblioteci
Python Tutor	vizualizarea pas cu pas a stării programului
medii online	rulare rapidă fără instalări complexe
vizualizatoare de algoritmi	urmărirea operațiilor și a structurilor de date
comunități profesionale	exemple, clarificări, schimb de resurse
microînvățare proprie	15–30 min focalizat pe următoarea secvență didactică

MESAJ SUBTIL	
	Nu este necesar ca profesorul să epuizeze limbajul Python înainte de prima lecție. Este necesar să stăpânească sigur conceptele și instrumentele pe care le folosește în secvența proiectată și să știe cum verifică ceea ce urmează să utilizeze.

30 REZERVA CURRICULARĂ DE 25%

Spațiu pentru răspuns la clasa reală, nu „timp rămas”.

SITUAȚIE	UTILIZARE
limbaje diferite în gimnaziu	armonizare Python prin probleme cunoscute
proiectarea cu dificultate a algoritmilor	remediere prin trasare și pseudocod
clasă avansată	aprofundare prin eficiență/comparație
prelucrarea cu dificultate a listelor	consolidare diferențiată
interes pentru proiect	extindere prin mini-aplicație

31 CUM CONSTRUIM PARCURSUL ANUAL LA SPECIALIZAREA MATEMATICĂ-INFORMATICĂ?

2 ore/săptămână: echilibru între modelare conceptuală, algoritmi și laborator.

Conținuturi	ORE ORIENTATIVE
Strategii de rezolvare a problemelor. Principii de elaborare a unui program	2
Strategii de rezolvare a problemelor. Prelucrări ale datelor numerice	10
Memorarea datelor și organizarea codului în limbaj de programare. Subprograme	8
Memorarea datelor și organizarea codului în limbaj de programare. Introducere în programarea orientată pe obiecte în limbaj de programare	1
Memorarea datelor și organizarea codului în limbaj de programare. Biblioteca Tkinter din Python pentru interfețe grafice	6
Memorarea datelor și organizarea codului în limbaj de programare. Fișiere text	4
Organizarea conceptuală a datelor. Modelul conceptual liniar – listă	3
Memorarea datelor și organizarea codului în limbaj de programare. Clasa list din Python	5
Strategii de rezolvare a problemelor. Metode de generare sistematică a elementelor unei liste	6
Strategii de rezolvare a problemelor. Metode de sortare a elementelor unei liste	6
Integrare și rezervă	16

NOTĂ	Orele sunt orientative și includ spațiu pentru reglaj; anexa Excel oferă modelul agregat, nu o planificare rigidă pe săptămâni.

32 CUM CONSTRUIM PARCURSUL ANUAL LA SPECIALIZAREA ȘTIINȚE ALE NATURII?

1 oră/săptămână: selectăm esențialul și folosim contexte din științe.

Conținuturi	ORE ORIENTATIVE
Strategii de rezolvare a problemelor. Principii de elaborare a unui program	2
Strategii de rezolvare a problemelor. Prelucrări ale datelor numerice	6
Memorarea datelor și organizarea codului în limbaj de programare. Subprograme	5
Memorarea datelor și organizarea codului în limbaj de programare. Introducere în programarea orientată pe obiecte în limbaj de programare	1
Memorarea datelor și organizarea codului în limbaj de programare. Fișiere text	2
Organizarea conceptuală a datelor. Modelul conceptual liniar – listă	1
Memorarea datelor și organizarea codului în limbaj de programare. Clasa list din Python	3
Strategii de rezolvare a problemelor. Metode de generare sistematică a elementelor unei liste	3
Strategii de rezolvare a problemelor. Metode de sortare a elementelor unei liste	3
Integrare și rezervă	8

DECIZIE	
	Nu importăm automat extensiile din programa de matematică-informatică sau matematică-informatică militară. La o oră pe săptămână, progresul competenței este mai important decât densitatea conținutului suplimentar.

33 CUM CONSTRUIM PARCURSUL ANUAL LA SPECIALIZAREA MATEMATICĂ-INFORMATICĂ MILITARĂ?

3 ore/săptămână: mai mult laborator și aprofundări specifice.

Conținuturi	ORE ORIENTATIVE
Strategii de rezolvare a problemelor. Principii de elaborare a unui program	2
Strategii de rezolvare a problemelor. Prelucrări ale datelor numerice	14
Memorarea datelor și organizarea codului în limbaj de programare. Subprograme	10
Memorarea datelor și organizarea codului în limbaj de programare. Introducere în programarea orientată pe obiecte în limbaj de programare	3
Memorarea datelor și organizarea codului în limbaj de programare. Biblioteca Tkinter din Python pentru interfețe grafice	10
Memorarea datelor și organizarea codului în limbaj de programare. Fișiere text	6
Organizarea conceptuală a datelor. Modelul conceptual liniar – listă	6
Memorarea datelor și organizarea codului în limbaj de programare. Clasa list din Python	10
Strategii de rezolvare a problemelor. Metode de generare sistematică a elementelor unei liste	9
Strategii de rezolvare a problemelor. Metode de sortare a elementelor unei liste	6
Integrare și rezervă	26

NOTĂ	Orele sunt orientative și includ spațiu pentru reglaj; anexa Excel oferă modelul agregat, nu o planificare rigidă pe săptămâni.

34 INTERDISCIPLINARITATE REALĂ

Problemele celorlalte discipline devin surse de date, modele și criterii.

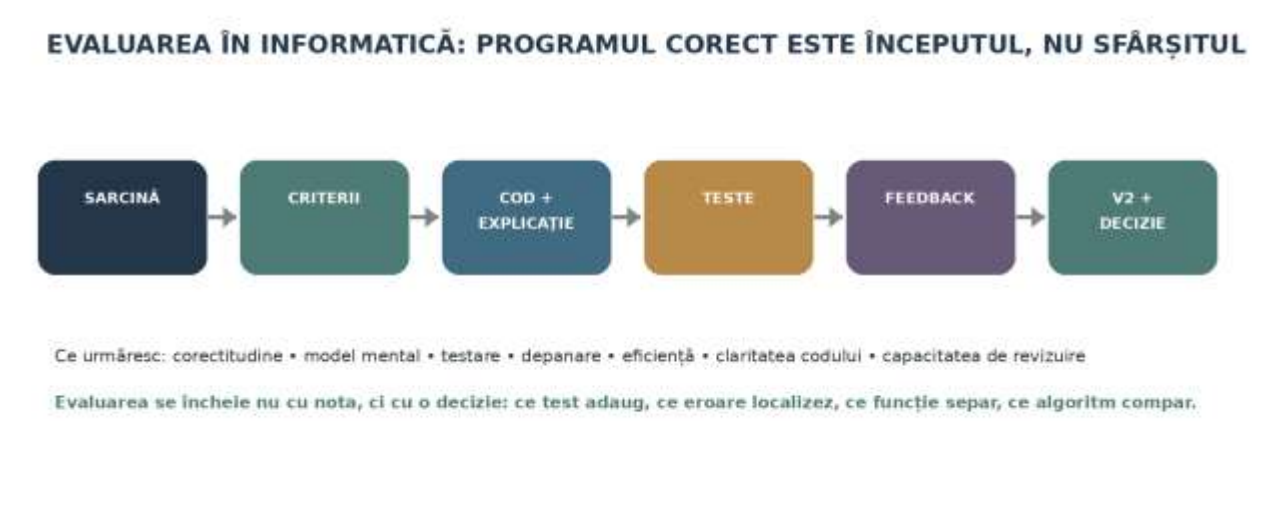
PARTENER	CONEXIUNE
matematică	divizibilitate, șiruri, funcții, eficiență
fizică	date experimentale, simulare, mișcare
chimie	prelucrare de date și calcule
biologie	serii de măsurători, clasificare simplă, date
geografie	serii temporale, indicatori, fișiere de date
economie	cheltuieli, tranzacții, sortări, frecvențe
arte digitale	interfață, vizualizare și produs digital

CHEIE	
	Interdisciplinaritatea este autentică atunci când contextul schimbă datele, modelarea și criteriile soluției, nu doar povestea din enunț.

35

EVALUAREA CARE PRODUCE ÎNVĂȚARE

În informatică, putem observa direct procesul de gândire prin cod, stare și teste.



DOVADĂ	CRITERIU
cod	corectitudine și claritate
trasare	model mental al executării
teste	acoperirea cazurilor relevante
depanare	localizarea și explicarea cauzei
comparație	argument privind alegerea și eficiența
V1→V2	capacitatea de revizuire

36 FEEDBACK CARE LASĂ ELEVULUI EFORTUL COGNITIV

Nu reparăm în locul elevului.

MAI PUȚIN UTIL	MAI UTIL
„Greșit. Vezi bucla.”	„Pentru n=1, bucla nu se execută. Verifică valorile produse de range și compară-le cu intervalul cerut.”
„Ai uitat ceva la listă.”	„După b=a, modificarea lui b se vede și în a. Explică de ce și decide dacă ai nevoie de o copie distinctă.”
„Soluția e lentă.”	„Ambele soluții sunt corecte. Estimează numărul de divizori testați pentru $n=10^6$ și justifică alegerea.”

EVALUAREA SE ÎNCHEIE	…nu cu nota, ci cu o decizie: ce test adaug, ce pas urmăresc, ce funcție separ, ce variantă compar, ce sprijin retrag.

37 DIFERENȚIERE FĂRĂ TREI PROGRAME PARALELE

Aceeași competență, sprijin diferit.

DIMENSIUNE	MAI MULT SPRIJIN	MAI MULTĂ AUTONOMIE
analiză	date și ieșire marcate	elevul extrage specificația
Python	schelet de cod	alege structura și funcțiile
liste	operații sugerate	selectează metodele
testare	cazuri oferite	proiectează cazurile
depanare	tabel de trasare	alege metoda de diagnostic
proiect	module date	descompune singur problema

38 DE LA PROGRAMĂ LA CLASĂ: DOUĂ MICROSECVENȚE

Fiecare leagă intenția, activitatea, dovada, feedbackul și revizuirea.

A. Programul greșit intenționat

MOMENT	ACTIVITATE	DOVADĂ
1	profesorul oferă un program Python cu eroare logică	predicție
2	elevii rulează un caz care nu descoperă eroarea	observație
3	proiectează un caz-limită	test
4	localizează primul pas greșit	trasare
5	corectează și păstrează testul în set	V2 + test de regresie

B. IA propune, elevul validează

MOMENT	ACTIVITATE	DOVADĂ
1	se cere IA o soluție la o problemă deja analizată	răspuns IA
2	elevul identifică algoritmul și facilitățile Python folosite	hartă
3	proiectează teste	suită de teste
4	compară cu propria soluție	argument
5	acceptă, modifică sau respinge soluția IA	decizie justificată

39 CE MAI PUTEM FOLOSI DIN REPERELE METODOLOGICE ELABORATE ÎN ANUL 2021?

Documentul este bogat și merită folosit selectiv, nu preluat mecanic.

SECȚIUNE	PAGINI	STATUT 2026–2027
continuitate gimnaziu–liceu	6–10	foarte utilă; actualizăm limbajul precizat în programă
evaluare inițială	11–31	utilă ca bancă; folosim și standardele 2026
construirea achizițiilor	32–62	utilă selectiv; actualizăm CS/conținuturi
adaptare elevi în risc	63–66	principii încă valabile
exemple paralele C++/Python	16–18 și alte secvențe	deosebit de utile ca precedent pentru transfer

RĂMÂNE VALABIL	
	Reperele metodologice elaborate în 2021 spuneau deja că elevii puteau veni din gimnaziu cu Python, C sau C++ și recomandau „armonizarea” limbajelor la intrarea în liceu. Noutatea 2025 este că Python devine limbajul obligatoriu de utilizat, conform programelor noi.

40

ÎNTREBĂRI PENTRU PROIECTAREA DIDACTICĂ

Un audit rapid înainte de o unitate sau de o resursă nouă.

- Ce CS urmăresc și ce dovadă o face vizibilă?
- Ce achiziție din gimnaziu presupun și am verificat-o?
- Elevul are o problemă algoritmică sau doar o problemă de sintaxă Python?
- Am ales un exemplu care clarifică un concept sau doar arată o facilitate de limbaj?
- Elevul prezice înainte să ruleze?
- Are de proiectat cazuri de test?
- Poate explica starea programului la un moment dat?
- Feedbackul indică primul pas relevant sau dă soluția?
- Am comparat soluții când există valoare cognitivă în comparație?
- Folosesc rezerva de 25% pentru nevoia reală a clasei?
- Dacă folosesc IA, am rulat și verificat codul?
- Resursa IA se raportează la CS și conținutul obligatoriu?
- Terminologia este clară în limba română?
- Profesorul/elevul poate explica independent soluția propusă de IA?
- Ce decizie urmează după evaluare?

41 DOCUMENTE DE REFERINȚĂ ȘI ANEXA EXCEL

Sursele curriculare utilizate și instrumentul de proiectare asociat.

- Ministerul Educației și Cercetării. (2025). Programa școlară pentru disciplina Informatică, clasa a IX-a, curriculum de specialitate, filiera teoretică, profil real, specializarea matematică-informatică. Anexa nr. 42 la OMEC nr. 6930/19.12.2025.
- Ministerul Educației și Cercetării. (2025). Programa școlară pentru disciplina Informatică, clasa a IX-a, curriculum de specialitate, filiera teoretică, profil real, specializarea științe ale naturii. Anexa nr. 43 la OMEC nr. 6930/19.12.2025.
- Ministerul Educației și Cercetării. (2025). Programa școlară pentru disciplina Informatică, clasa a IX-a, curriculum de specialitate, filiera vocațională, profil militar, specializarea matematică-informatică militară. Anexa nr. 66 la OMEC nr. 6930/19.12.2025.
- Ministerul Educației, Cercetării și Inovării. (2009). Programa școlară pentru Informatică, clasa a IX-a, aprobată prin OMECI nr. 5099/09.09.2009.
- Ministerul Educației Naționale. (2017). Programa școlară pentru disciplina Informatică și TIC, clasele a V-a–a VIII-a, aprobată prin OMEN nr. 3393/28.02.2017.
- Ministerul Educației și Cercetării – Centrul Național pentru Curriculum și Evaluare. (2026). Standarde naționale de evaluare asociate disciplinei Informatică și TIC, clasele a V-a–a VIII-a, Anexa nr. 15.
- Ministerul Educației – Centrul Național de Politici și Evaluare în Educație. (2021). Repere metodologice pentru aplicarea curriculumului la clasa a IX-a – Informatică.

ANEXĂ

PLANIFICAREA CU SENS – INFORMATICĂ, CLASA a IX-a, CURRICULUM DE SPECIALITATE, 2026–2027. Registrul conține separat cele trei programe, CG/CS/EAI și conținuturile integrale, corespondențe CS–conținuturi, trei planificări agregate, standardele VIII, harta C++ → Python, traseul de 10 ore pentru profesor, utilizări ale IA, verificarea resurselor IA, strategii, evaluare și rezerva de 25%.

42	**NOTĂ PRIVIND REALIZAREA DOCUMENTULUI** Transparență privind utilizarea IA și drepturile asupra produsului.

TRANSPARENȚĂ	Prezentul material a fost elaborat cu sprijinul unui asistent bazat pe inteligență artificială, utilizat ca instrument de documentare, structurare, redactare și revizuire, cu respectarea cadrului legal aplicabil și a regulilor privind elaborarea materialelor educaționale. Conținutul final a fost analizat, selectat, adaptat și validat în cadrul procesului de elaborare, constituind un produs original. Drepturile asupra formei finale a materialului aparțin Ministerului Educației și Cercetării – Centrului Național pentru Curriculum și Evaluare (MEC–CNCE).
NOTĂ CURRICULARĂ	Reperele au caracter orientativ și nu limitează autonomia profesională. Programele școlare sunt documentele de referință obligatorii. Diferențele dintre cele trei programe sunt păstrate; standardele gimnaziale sunt utilizate numai pentru diagnoza de intrare; pentru clasa a IX-a nu sunt elaborate standarde.

Schimbarea limbajului de programare poate fi tratată ca o ocazie de a face mai vizibil ceea ce profesorul știe deja foarte bine: să transforme o problemă într-un algoritm, să urmărească executarea, să testeze o soluție și să învețe din eroare.