MINISTERUL EDUCAȚIEI
CENTRUL NAȚIONAL DE POLITICI ȘI EVALUARE ÎN EDUCAȚIE

REPERE METODOLOGICE

PENTRU APLICAREA CURRICULUMULUI
LA CLASA a IX-a
ÎN ANUL ȘCOLAR 2021-2022

Disciplina INFORMATICĂ

filiera teoretică, profilul real, specializările matematică-informatică, ştiinţe ale naturii şi matematică-informatică intensiv informatică
filiera vocaţională, profilul militar, specializarea matematică-informatică

București, 2021

CUPRINS

INTRODUCERE

Aceste repere metodologice sunt realizate cu scopul de a veni în sprijinul profesorilor de informatică pentru organizarea și derularea demersului didactic la clasa a IX-a, la disciplina informatică, filiera teoretică, profilul real, specializările matematică-informatică și științe ale naturii, respectiv matematică-informatică intensiv informatică, și filiera vocațională, profilul militar, specializarea matematică-informatică.

"O educaţie bine făcută poate întotdeauna să scoată dintr-un suflet, oricare ar fi el, partea folositoare pe care o conţine." (Victor Hugo)

Disciplina informatică își aduce aportul la formarea și dezvoltarea capacității de rezolvare a problemelor apărute în diverse situații în mod coerent, logic, rațional, având în vedere nu doar un rezultat corect, ci și o utilizare eficientă a resurselor, din toate punctele de vedere. Primii pași în formarea gândirii algoritmice au fost realizați prin studiului disciplinei informatică și TIC, pe parcursul ciclului gimnazial, pași care stau la baza proceselor de înțelegere și aprofundare necesare pe parcursul următorilor ani de studiu.

De la curriculumul scris la cel aplicat

Curriculumul naţional scris (intenționat), conceput pe baza unui ansamblu de principii, asigură flexibilizarea şi personalizarea demersului didactic, în acord cu nevoile, interesele şi ritmurile diferite de dezvoltare a elevilor.

Profesorul are libertatea contextualizării programei şcolare şi proiectării unor parcursuri de învățare personalizate. Proiectarea demersului didactic se realizează prin raportare la programa şcolară şi presupune:

- lectura integrală şi personalizată a programei școlare;
- elaborarea planificării calendaristice;
- proiectarea unităților de învățare.

Astfel, la nivelul **curriculumului aplicat**, diversitatea contextelor conduce la o diversitate a abordărilor materializate într-o multitudine de parcursuri ale programelor școlare.

Reperele metodologice vor oferi sprijin profesorilor în trecerea de la curriculumul scris (intenționat) la cel aplicat (implementat). Identificarea discontinuităților dintre achizițiile învățării la finalul ciclului gimnazial și achizițiile, așteptate și necesare, pentru debutul clasei a IX-a, dar și

diminuarea diferențelor, la nivelul implementării curriculumului, prin propunerea de soluții inovatoare, va asigura un debut liceal eficient.

Competenţele specifice precizate în programa școlară în vigoare sunt derivate din competenţele generale și reprezintă etape în dobândirea acestora, formându-se pe durata unui an şcolar.

Conţinuturile învăţării precizate în programa școlară în vigoare sunt acele mijloace informaţionale care reprezintă baza de operare pentru formarea competenţelor.

Curriculumul scris, corelat cu implicarea profesorilor în alegerea activităților de învățare adecvate, în strânsă legătură cu contextul generat de clasele de elevi și adaptate la particularitățile acestora, va asigura progresul învățării prin trecerea de la un an de studiu la altul.

Pentru o proiectare și o planificare corespunzătoare a activității didactice, fiecare profesor trebuie să pornească de la analiza competențelor dobândite de elevi pe parcursul ciclului gimnazial la disciplina informatică și TIC și să țină cont de contextul cauzat de pandemia COVID-19 și de avantajele/dezavantajele rezultate din combinațiile studiului sincron și asincron, derulat la distanță, cu studiul disciplinei în clasă.

În vederea reuşitei demersului didactic, profesorul trebuie să identifice, la începutul clasei a IX-a, eventualele lacune în achizițiile elevilor, și să ia măsurile care se impun, astfel încât să reușească, pe parcursul anului școlar, să consolideze competențele formate și să dezvolte altele noi, prin parcurgerea conținuturilor precizate în programă la nivelul de vârstă adecvat elevilor de liceu.

Fiind primul an în care programa școlară pentru clasa a IX-a se adresează unor elevi care au studiat în gimnaziu disciplina informatică și TIC, se recomandă ca proiectarea demersului didactic ce vizează formarea competențelor specifice să urmărească valorificarea achizițiilor elevilor de pe parcursul ciclului gimnazial.

Astfel, profesorul de informatică va stabili conexiunile dintre competențele formate pe parcursul ciclului gimnazial și competențele specifice precizate în programa școlară a disciplinei pentru clasa a IX-a, concentrându-se pe fuziunea acestora: ce trebuie recapitulat și fixat corespunzător, cum trebuie armonizate competențele și conținuturile pentru a asigura continuitatea și coerența între ciclul gimnazial și cel liceal.

Conform sugestiilor metodologice din programa școlară de informatică și TIC pentru gimnaziu la clasele a VII-a și a VIII-a pentru formarea competențelor specifice s-a putut utiliza un limbaj de programare dintr-o listă care cuprinde, de exemplu propuneri ca Python, C, C++. Deoarece alegerea limbajului de programare a fost la latitudinea profesorului, este necesară o armonizare a acestor limbaje de programare acum, la trecerea în clasa a IX-a, pentru a utiliza unul

dintre limbajele precizate în programa școlară în vigoare (dintre acestea, recomandându-se limbajul C++).

Contextul actual impune o adaptare metodologică, reflectată prin modalităţile de organizare a conţinuturilor, metodelor de predare-învăţare-evaluare, a mijloacelor didactice, a mediului psihologic de învăţare în scopul diferenţierii experienţelor de învăţare şi de adaptare a procesului instructiv-educativ la posibilităţile de înţelegere, la ritmul şi la stilul de învăţare ale elevului.

Un loc aparte trebuie să-l constituie îmbinarea și armonizarea metodelor didactice tradiționale cu cele moderne, atât la evaluarea gradului de formare a competențelor specifice ciclului gimnazial, cât și la formarea competențelor specifice în clasa a IX-a și la dezvoltarea personalităţii elevilor.

Metodele tradiţionale trebuie permanent adaptate la exigenţele unui învăţământ modern, promovând centrarea pe elev, învăţarea prin descoperire, autoevaluarea. Astfel, elevii trebuie să devină participanți activi la educaţie, să fie încurajați să învețe, să fie pregătiți să urmărească să rezolve orice problemă care apare pe parcursul învățării.

În activitatea la clasă pot fi integrate metode didactice moderne de dezvoltare a gândirii critice ca „Turul galeriei”, „Mozaic”, „Cubul”, „Gândiţi, lucraţi în echipă, comunicaţi!”, „Termeni-cheie iniţiali”, „Sinelg”, elaborarea unui referat/eseu, tehnica predicţiei, învăţarea în grupuri mici, „Turneul” între echipe, „Hărțile conceptuale”.

Întrucât în centrul demersului didactic este plasat elevul, ca subiect al activității instructiv-educative, se recomandă ca lecțiile să valorifice curiozitatea și interesele de cunoaștere și învățare ale elevilor, astfel încât să-i pună pe aceștia în situația de a se implica activ în propria formare și dezvoltare.

În acest material sunt propuse exemple de itemi și de teste sau exemple de aplicare a unor metode moderne care vizează evaluarea inițială, dar și exemple de activități de învățare remediale corespunzătoare.

De asemenea, sunt prezente exemple de activităţi de învăţare - sarcini de lucru cu ajutorul cărora se formează şi se dezvoltă competenţele specifice pentru clasa a IX-a și care pot integra strategii didactice adecvate unor contexte de învăţare variate.

Materialul prezintă și unele recomandări privind modul de abordare a conținuturilor precizate în programă printr-un demers didactic adaptat la particularitățile/categoriile elevilor aflați în situații de risc.

Documente școlare necesare:

- programa școlară pentru disciplina informatică, clasa a IX-a, filiera teoretică, profil real, specializarea matematică-informatică intensiv informatică, aprobată prin O.M.nr.5099/09.09.2009 (http://programe.ise.ro/Portals/1/Curriculum/Progr_Lic/TH/Informatica_teoretic_vocational_intensiv_clasa%20a%20IX-a.pdf);
- programa școlară pentru disciplina informatică, clasa a IX-a, filiera teoretică, profil real, specializările matematică-informatică și științe ale naturii, respectiv filiera vocațională, profil militar, specializarea matematică-informatică, aprobată prin O.M. nr. 5099/09.09.2009 (http://programe.ise.ro/Portals/1/Curriculum/Progr_Lic/TH/Informatica_teoretic_vocational_clasa%20a%20IX-a.pdf);
- programa școlară pentru disciplina informatică și TIC, clasele V-VIII, aprobată prin O.M.nr.3393/2017 (programa pentru gimnaziu: (http://programe.ise.ro/Portals/1/Curriculum/2017-progr/117-INFORMATICA%20si%20TIC.pdf).

Repere și resurse educaționale:

- repere metodologice la disciplina informatică și TIC pentru gimnaziu https://educatiacontinua.edu.ro/repere-metodologice.html;
- resurse educaționale pentru elevi și profesori https://educatiacontinua.edu.ro/resurse-educationale.html;
- instrumente de lucru pentru elevi https://educatiacontinua.edu.ro/instrumente-de-lucru-pentru-elevi.html.

1. PLANIFICAREA CALENDARISTICĂ

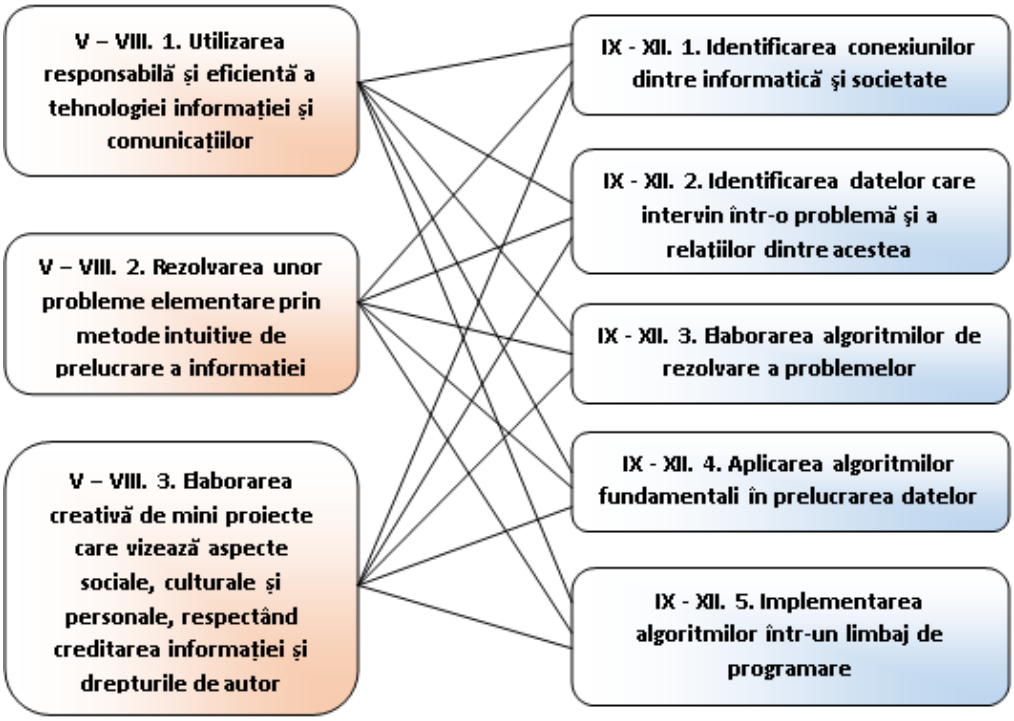
Planificarea calendaristică este un document de proiectare curriculară care asociază elemente ale programei școlare cu alocarea de timp considerată optimă de către profesor pe parcursul unui an şcolar.

Recomandări pentru realizarea planificării calendaristice

Planificarea calendaristică la disciplina informatică, pentru clasa a IX-a, anul şcolar 2021 – 2022, trebuie să ţină cont de faptul că absolvenţii clasei a VIII au sau ar trebui să aibă achiziţii specifice disciplinei. Pornind de la această premisă, realizarea planificării calendaristice se face parcurgând următoarele etape:

Etapa 1. Consultarea programelor pentru clasele a V-a – a VIII-a, dar şi a programei pentru clasa a IX-a în scopul identificării punctelor comune, a ancorelor care pot sta la baza demersului didactic în noul an şcolar.

La nivelul **competenţelor generale** formate/dezvoltate pe parcursul studiilor liceale există o continuitate, precum şi o nuanţare a competenţelor generale formate pe parcursul ciclului gimnazial, aşa cum este ilustrat mai jos.

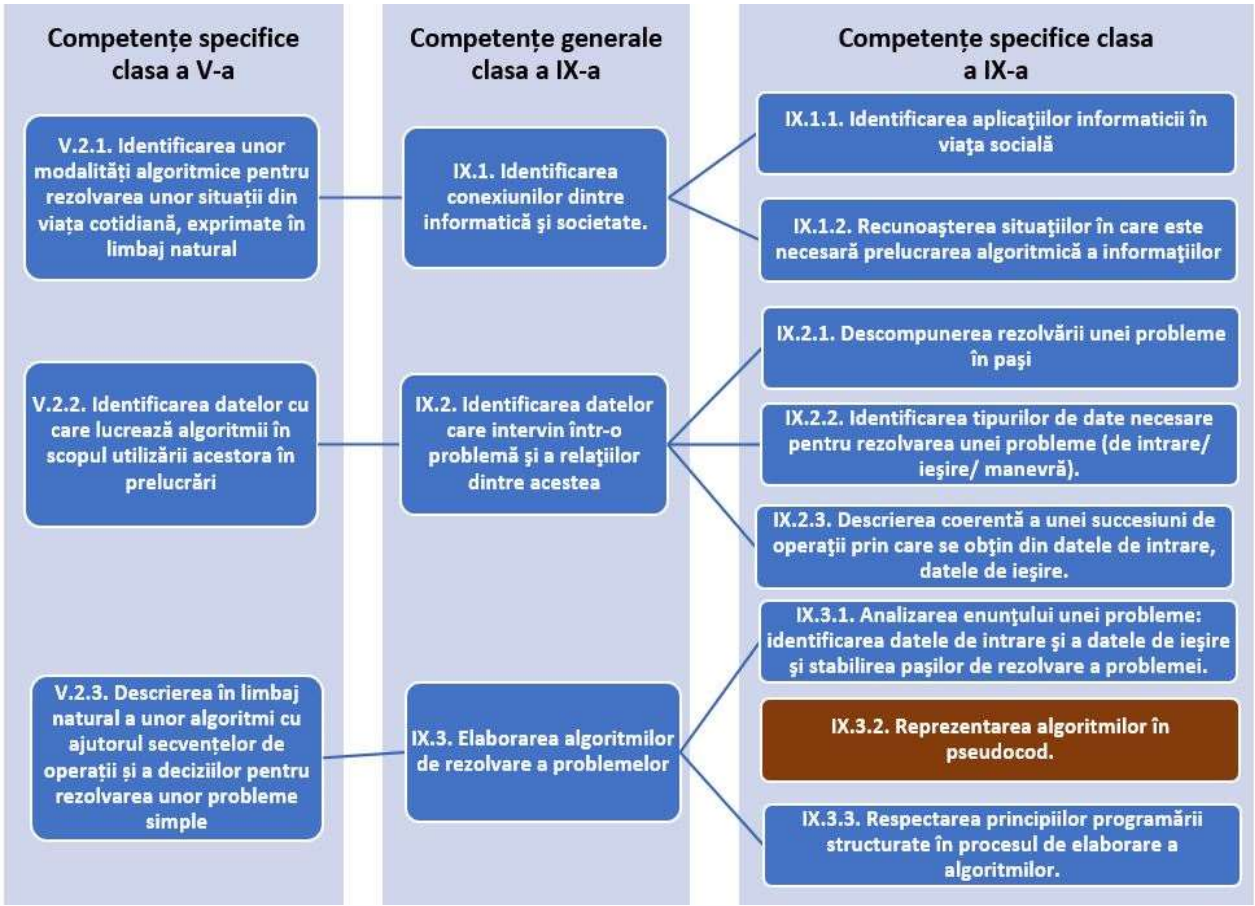


Competența generală ***1. Utilizarea responsabilă și eficientă a tehnologiei informației și comunicațiilor*** formată pe parcursul ciclului gimnazial, prin competența specifică ***1.4 Utilizarea unui mediu de programare pentru implementarea algoritmilor*** (clasa a VII-a), poate fi pusă în corespondență cu competența generală ***5. Implementarea algoritmilor într-un limbaj de programare***, formată pe parcursul ciclului liceal.

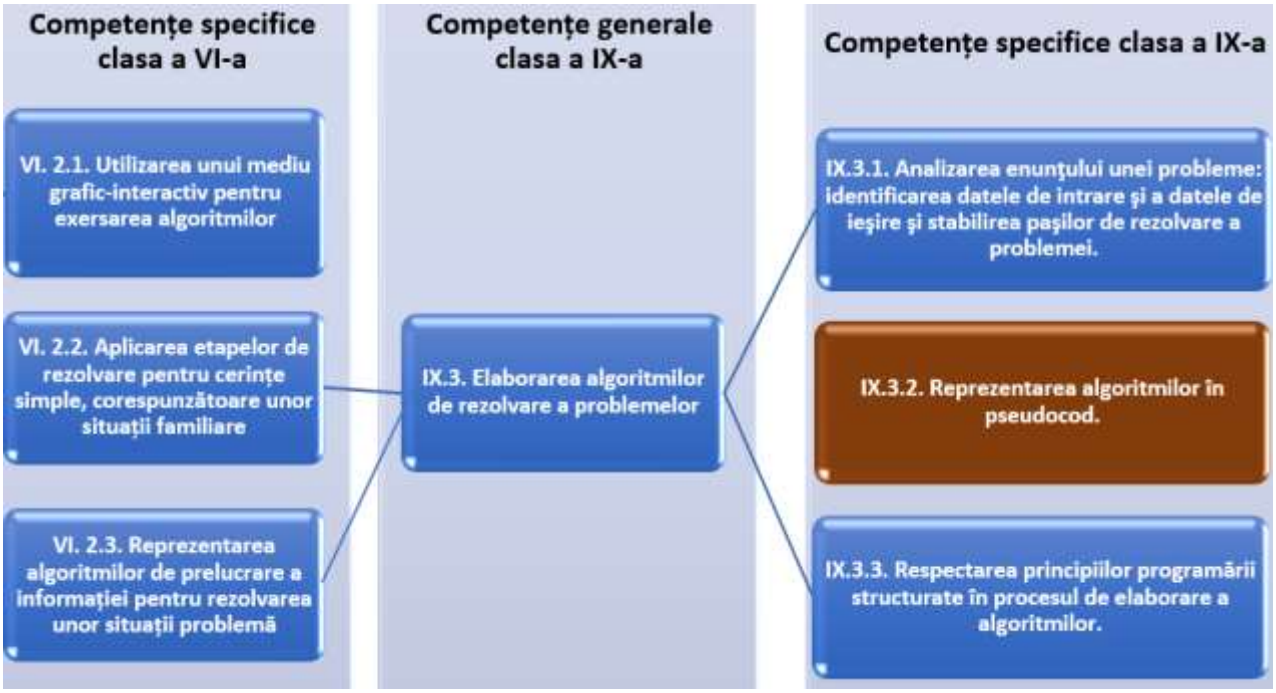
Competența generală ***3. Elaborarea creativă de mini proiecte care vizează aspecte sociale, culturale și personale, respectând creditarea informației și drepturile de autor*** formată pe parcursul ciclului gimnazial, prin competența specifică ***3.3. Implementarea algoritmilor într-un mediu de programare*** (clasa a VIII-a), poate fi pusă în corespondență cu ***5. Implementarea algoritmilor într-un limbaj de programare***, formată pe parcursul ciclului liceal**.**

Particularizând analiza pentru competența generală ***2. Rezolvarea unor probleme elementare prin construirea unor algoritmi de prelucrare a informației*** formată pe parcursul ciclului gimnazial, se observă următoarele corespondențe cu competențele generale formate pe parcursul ciclului gimnazial, respectiv cu competențele specifice formate pe parcursul clasei a IX-a:

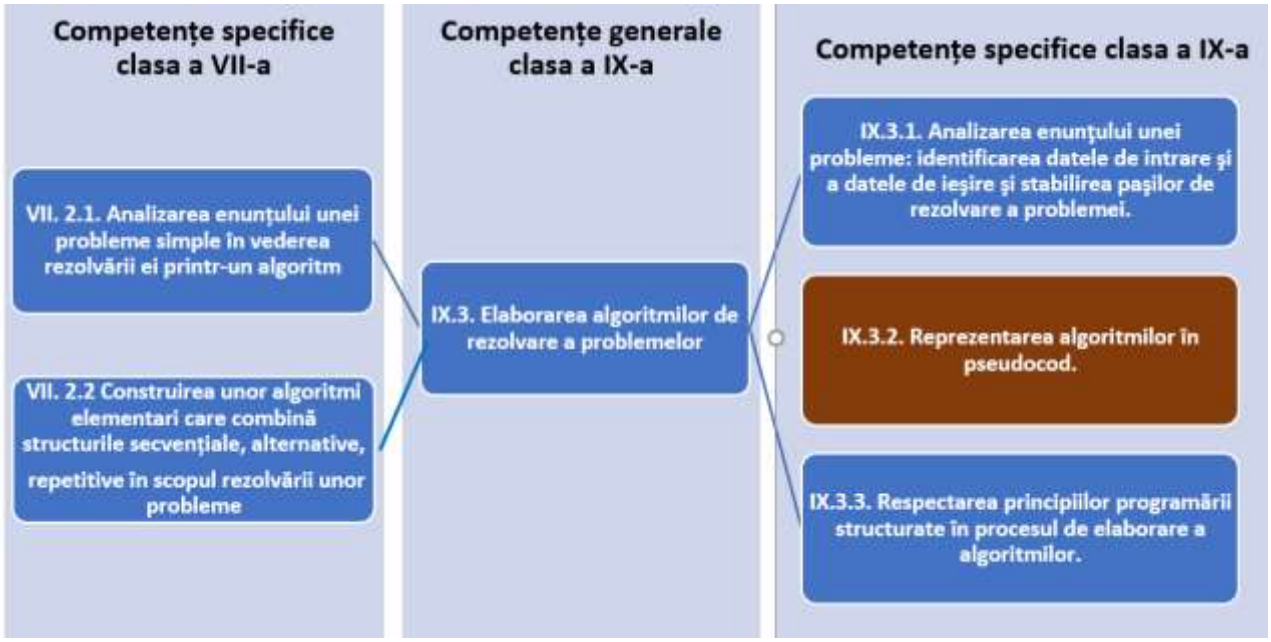
- Clasa a V-a



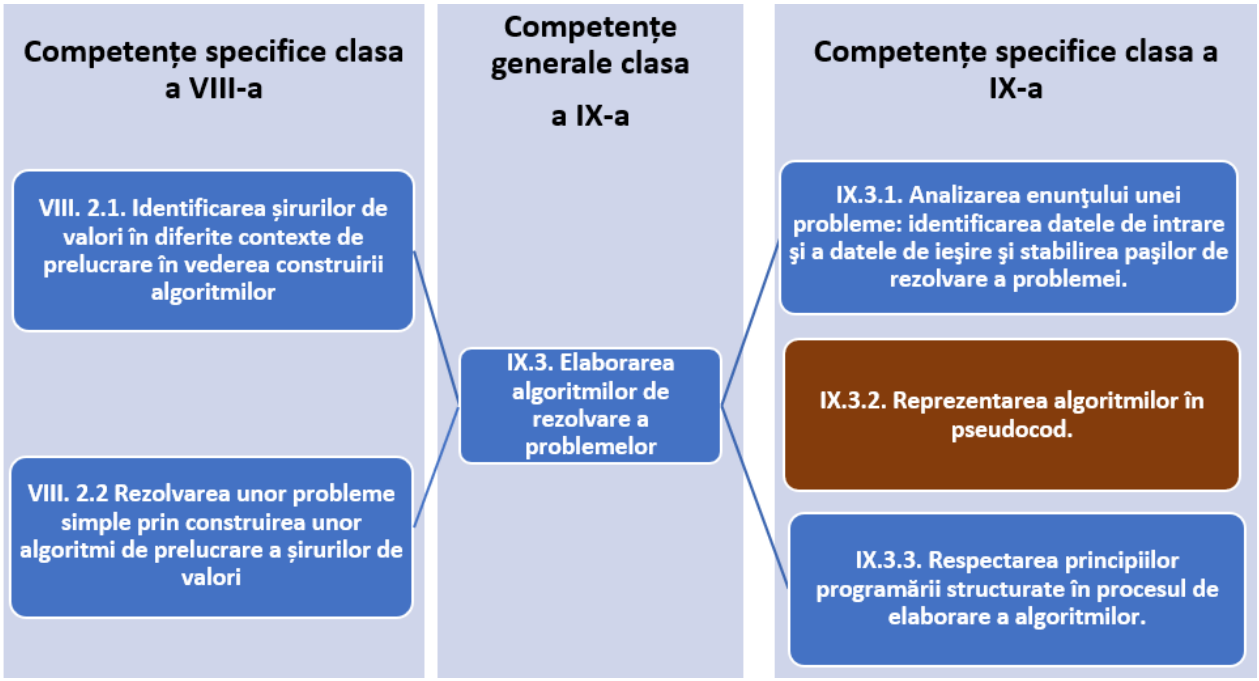
- Clasa a VI-a



- Clasa a VII-a



- Clasa a VIII-a



Vor fi avute în vedere elemente specifice disciplinei, de exemplu:

- consolidarea achizițiilor din gimnaziu prin reluarea și abordarea unor anumite elemente de conținut la un nivel corespunzător vârstei, în clasa a IX-a;
- valorificarea competențelor formate pe parcursul ciclului gimnazial (de exemplu implementarea algoritmilor în limbaje de programare, inclusiv în cadrul specializărilor matematică-informatică și științe ale naturii), pentru a asigura continuitatea procesului de predare-învățare;
- armonizarea tehnologiilor suport pentru învățare (de exemplu editoare pentru editarea și executarea programelor, limbaje de programare), utilizate pe parcursul ciclului gimnazial, printr-o abordare comparativă, în vederea familiarizării elevilor cu cele utilizate la liceu, în funcție de specificul unității școlare și corespunzător programei de clasa a IX-a.

Etapa a 2-a. Alocarea timpului și stabilirea activităților didactice specifice pentru determinarea nivelului achiziţiilor (exprimate în termeni de competenţe) pe care le deţin elevii.

Evaluarea inițială reprezintă un factor cheie în determinarea ”punctului de plecare” atât al procesului de învățare, cât și al planificării anuale și semestriale.

Eventuale lacune în achiziţiile existente pot reprezenta premisele unor dificultăţi în formarea de noi competenţe sau în dezvoltarea unor competenţe deja formate, așa cum identificarea elementelor temeinic asimilate pot fi ulterior exploatate la un nivel superior.

Astfel, în această etapă, profesorul va realiza conexiuni între componentele estimate a fi insuficient asimilate/neasimilate și programa pentru clasa a IX-a.

Ca urmare, se recomandă includerea unei perioade de două-trei săptămâni de evaluări inițiale temeinice și de activităţi recapitulative, care vor viza acele competenţe, formate pe parcursul întregului ciclu gimnazial, menite să permită o conexiune clară între achiziţiile existente şi cele prevăzute de programa pentru clasa a IX-a.

În această etapă se vor clarifica întrebări ca: „Ce achiziţii, au elevii?”, „Care este nivelul acestor achiziţii?”, „Care sunt elementele pe care se poate construi?”, „Care sunt secvenţele curriculare ce necesită acţiuni de remediere/recuperare?”. Pe baza constatărilor făcute în această etapă se vor construi demersurile didactice viitoare și, eventual, se actualizează/ajustează planificarea didactică.

Etapa a 3-a. Ajustarea planificării iniţiale pe baza celor constatate în urma evaluării.

În acest moment se poate stabili cu mai mare exactitate bugetul de timp alocat fiecărei unităţi de învăţare şi defalcarea acestuia pe activităţi de învăţare.

Considerând evaluarea inițială ca punct de reper, dacă este cazul, profesorul realizează un plan de recuperare și proiectează activitatea diferențiată, proiectând activităţi destinate elevilor cu nevoi speciale şi a celor cu rezultate aflate la extreme: elevii cu rezultate foarte scăzute şi elevii cu rezultate foarte ridicate.

Odată realizată planificarea calendaristică se poate trece la proiectarea unităţilor de învăţare, cu enumerarea activităţilor de învăţare ce urmează a fi desfăşurate în vederea formării competenţelor specifice.

2. EVALUAREA GRADULUI DE ACHIZIȚIE A COMPETENȚELOR DIN GIMNAZIU

2. A. Recomandări pentru realizarea evaluării inițiale

Procesul de învățământ este reprezentat de activitatea instructiv-educativă care cuprinde trei componente: predare, învățare, evaluare, aflate în relații de interdependență. Evaluarea oferă profesorului o privire de ansamblu asupra acțiunilor legate de predare și învățare, astfel încât să poată fi menținută direcția corectă în ceea ce privește formarea competențelor specifice. Se remarcă trei tipuri de evaluări: inițială/predictivă, continuă/formativă și finală/sumativă.

În această secțiune este abordată evaluarea inițială, care se impune la debutul clasei a IX-a pentru a sprijini cadrul didactic în aflarea răspunsurilor la întrebări de tipul:

- Planificarea calendaristică realizată este corespunzătoare colectivului de elevi sau necesită modificări?
- Care sunt achizițiile reale ale elevilor dobândite în cadrul disciplinei informatică și TIC pe parcursul celor patru ani de gimnaziu?
- Se impune o abordare prin activități de învățare remediale?
- Cum se realizează diferențierea? Ce nevoi au elevii aflați în situații de risc? Ce nevoi au elevii cu performanțe superioare?

Evaluarea inițială nu are rolul de a ierarhiza elevii, ci de a stabili punctul de plecare în formarea de noi achiziții, prin stabilirea nivelului curent de competențe. De aceea, profesorul trebuie să se asigure că:

- evaluarea inițială are loc imparțial;
- modalitatea de evaluare este comunicată în prealabil elevilor;
- în urma evaluării fiecare elev va fi încurajat să emită o autoevaluare obiectivă;
- fiecare elev va intra în posesia feedback-ului care să îi ofere în mod clar situația sa actuală în ceea ce privește procesul de instruire din gimnaziu.

Etapele pe care fiecare profesor le va parcurge în vederea realizării evaluării inițiale sunt evidențiate în cele ce urmează:

- stabilirea competențelor specifice care urmează să fie evaluate;
- selectarea metodei de evaluare care poate fi tradițională (probă scrisă, probă practică sau probă orală) sau modernă/alternativă, precum și a instrumentului de evaluare adecvat (test, chestionar);
- aplicarea instrumentului de evaluare în condiții stabilite și aduse la cunoștință elevilor;
- analizarea rezultatelor;
- comunicarea rezultatelor către elevi și părinți;
- luarea deciziilor.

În secțiunea anterioară au fost evidențiate relațiile dintre competențele specifice precizate în programa clasei a IX-a pentru disciplina informatică (atât pentru clasele cu specializarea matematică-informatică și științe ale naturii, cât și pentru clasele cu specializarea matematică-informatică intensiv informatică) și bagajul de competențe specifice care ar trebui să însoțească absolventul de gimnaziu.

Se recomandă utilizarea itemilor de tipuri diferite și cu grade de dificultate diferite. În această secțiune s-au avut în vedere trei niveluri cognitive/de complexitate:

- Nivelul 1 (de cunoaștere) – cunoștințe declarative, cunoștințe procedurale, cunoștințe contextuale; nivelul este evidențiat prin următoarele tipuri de sarcini: reamintirea informațiilor relevante, descrierea/exprimarea cu propriile cuvinte, exemplificarea, demonstrarea cunoștințelor în legătură cu utilizarea aparatelor, echipamentelor, instrumentelor;
- Nivelul 2 (de aplicare) – aplicarea cunoștințelor și înțelegerea conceptuală manifestată în situații problemă; nivelul este evidențiat prin următoarele tipuri de sarcini: comparare/diferențiere, relaționare, utilizare de modele, interpretare, explicare;
- Nivelul 3 (de raționament) – analizarea unor situații nefamiliare, a unor contexte complexe, formularea de concluzii și explicații, luarea deciziilor, transferul de cunoștințe în situații noi sau rezolvarea unor probleme ce presupun identificarea unei strategii de lucru; nivelul este evidențiat prin următoarele tipuri de sarcini: analiză, sinteză, formulare de întrebări/ipoteze/predicții, design/proiectare a investigațiilor, evaluare, justificare a concluziilor.

Pentru evaluarea inițială, sunt prezentate în continuare exemple de itemi, precum și un exemplu de test, ca instrument ce poate fi utilizat în cadrul unei metode tradiționale de evaluare, și exemple de aplicare a unor metode moderne de evaluare. Pentru testul corespunzător metodei tradiționale de evaluare este prezentă matricea de specificații, iar pentru ambele metode de evaluare sunt prezenți itemii corespunzători, baremul de evaluare și de notare, precum și propuneri de activități remediale.

Pentru itemii care vizează și elemente de limbaj de programare, se recomandă ca testul inițial să abordeze toate limbajele studiate pe parcursul gimnaziului de elevii clasei cărora li se adresează, urmând ca activitatea de predare-învățare-evaluare corespunzătoare competențelor și conținuturile aferente să aibă în vedere limbajul de programare precizat în programa în vigoare pentru clasa a IX-a.

2. B. Exemple de metode aplicate pentru evaluarea inițială

2. B. 1. Metodă tradițională – probă scrisă

Specializările matematică-informatică, științe ale naturii

Tipul probei:

scrisă

Timpul alocat pentru rezolvare:

45 de minute

Proiectarea testului:

Matricea de specificații:

Competențe generale și specifice din ciclul gimnazial evaluate	*Nivelul 1 (cunoaștere)*	*Nivelul 2 (aplicare)*	*Nivelul 3 (raționament)*
CG2. Rezolvarea unor probleme elementare prin metode intuitive de prelucrare a informațiilor. CS V.2.2. Identificarea datelor cu care lucrează algoritmii în scopul utilizării acestora în prelucrări.		II.1	

CG2. Rezolvarea unor probleme elementare prin metode intuitive de prelucrare a informațiilor. CS VI.2.3. Reprezentarea algoritmilor de prelucrare a informației pentru rezolvarea unor situații problemă.	I.2		
CG2. Rezolvarea unor probleme elementare prin metode intuitive de prelucrare a informațiilor. CS VII.2.1. Analiza enunțului unei probleme simple în vederea rezolvării ei printr-un algoritm.		II.3	
CG3. Elaborarea creativă de mini proiecte care vizează aspecte sociale, culturale și personale, respectând creditarea informației și drepturile de autor. CS VII.3.3. Implementarea algoritmilor într-un mediu de programare în scopul rezolvării creative a unor probleme având caracter aplicativ.	I.1		
CG3. Elaborarea creativă de mini proiecte care vizează aspecte sociale, culturale și personale, respectând creditarea informației și drepturile de autor. CS VII.3.3. Implementarea algoritmilor într-un mediu de programare în scopul rezolvării creative a unor probleme având caracter aplicativ.			II.5
CG2. Rezolvarea unor probleme elementare prin metode intuitive de prelucrare a informațiilor. CS VIII.2.1. Identificarea șirurilor de valori în diferite contexte de prelucrare în vederea construirii algoritmilor.			II.2
CG2. Rezolvarea unor probleme elementare prin metode intuitive de prelucrare a informațiilor. CS VIII.2.2. Rezolvarea unor probleme simple prin construirea unor algoritmi de prelucrare a șirurilor de valori.		II.4	

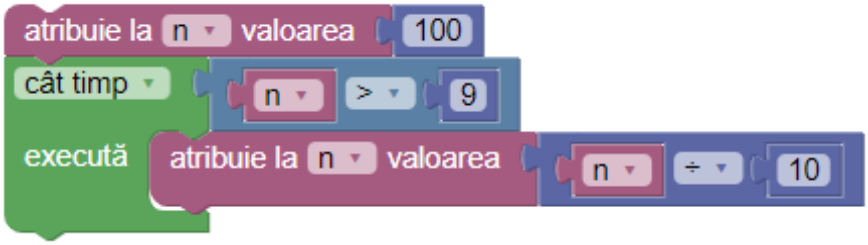
Cerințe:

I. Pentru cerințele numerotate cu **1.** și **2.** scrieți pe foaia de test litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu câte 10 puncte.

1. Identificați un simbol prin care poate fi reprezentat, în limbajul studiat, operatorul de înmulțire a două numere.

a. `&` **b.** `*` **c.** `•` **d.** `x`

2. Indicați structurile de control utilizate în secvența alăturată, în ordinea/relația corespunzătoare.



a. o structură secvențială (liniară), urmată de o structură alternativă (decizională) și de o structură repetitivă cu test final

b. o structură repetitivă cu test final, urmată de o structură secvențială (liniară)

c. o structură secvențială (liniară), urmată de o structură repetitivă cu test inițial care conține o structură secvențială (liniară)

d. o structură secvențială (liniară), urmată de o structură alternativă (decizională) care conține o structură secvențială (liniară)

II. Pentru cerințele următoare, scrieți pe foaia de test răspunsul corect.

1. Pentru cerința de mai jos, scrieți datele constante și datele variabile utilizate.

La începutul fiecărei vacanțe, mama pregătește o prăjitură. Pentru a face prăjitura, ea are nevoie de `5` ouă, `150` de grame de făină, `200` de grame de zahăr și `500` de grame de fructe. Se cere scrierea unui algoritm care să determine, de fiecare dată când mama face prăjitura, costul total al ingredientelor necesare, știind că un ou costă `x` lei, un kilogram de făină costă `y` lei, un kilogram de zahăr costă `z` lei și un kilogram de fructe costă `t` lei. **(10p.)**

2. Pentru fiecare dintre cerințele enumerate în coloana din stânga (**A**, **B**), asociați unul dintre algoritmii enumerați în coloana din dreapta (**1**, **2**, **3**, **4**, **5**) care ar putea fi folosit pentru rezolvare. Fiecărei cerințe din coloana din stânga îi corespunde un singur algoritm în coloana din dreapta.

A. Se cunosc numărul de elevi dintr-o clasă, precum și mediile generale ale acestora la sfârșitul unui an școlar. Se cere numărul elevilor care vor lua premiul I.	**1**. Algoritmul de determinare a celui mai mare divizor comun a două numere.
B. O firmă care asigură transporturi are două tipuri de camioane, care pot fi încărcate cu maximum **x**, respectiv maximum **y** tone de marfă la un transport. Știind că numărul total de transporturi a fost **n** și că s-au transportat în total **m** tone de marfă, se cere numărul de camioane din fiecare tip care au participat la această acțiune. Valorile notate cu **x**, **y**, **n** și **m** sunt întregi.	**2**. Nu se poate folosi un algoritm pentru rezolvare, problema este nedeterminată. **3.** Algoritmul de rezolvare a unui sistem de două ecuații cu două necunoscute. **4.** Algoritmul de calcul al numărului de divizori ai unui număr. **5.** Algoritmul de prelucrare a unui șir de numere care determină valoarea maximă din șir și numărul de apariții ale acesteia.

Scrieți asocierile determinate sub forma: literă-cifră. **(10p.)**

3. În secvența de program de mai jos, scrisă în limbaj de programare, toate variabilele sunt de tip real.

```
//limbajul C++
cin>>a;
cin>>b;
cin>>c;
cin>>d;
maxim=a;
if(...)
  maxim=b;
if(...)
  maxim=c;
...(maxim<d)
  maxim=d;
cout<<maxim;
```

```
#limbajul Python
a=float(input())
b=float(input())
c=float(input())
d=float(input())
maxim=a
if ...:
   maxim=b
if ...:
   maxim=c
...maxim<d:
   maxim=d
print(maxim)
```

Scrieți expresiile/cuvintele cheie ce pot înlocui punctele de suspensie, astfel încât, în urma executării secvenței obținute, să se afișeze pe ecran cea mai mare dintre cele patru valori reale citite. **(10p.)**

4. În secvența de program de mai jos, scrisă în limbaj de programare, toate variabilele sunt de tip întreg.

```
//limbajul C++
cin>>n;
k=0;
for(i=1;i<=n;i++)
{  cin>>x;
   if(x>=0 && x<10)
     k=k+1;
}
cout<<k;
```

```
#limbajul Python
n=int(input())
k=0
for i in range (1,n+1):
  x=int(input())
  if x>=0 and x<10:
     k=k+1
print(k)
```

Dacă, la executarea secvenței, pentru **n** se citește valoarea **5**, iar pentru **x** valorile **12, 1, 5, 9** și **22,** scrieți valorile variabilelor **k, i, x** la fiecare iterație, precum și valoarea afișată.

(20p.)

5. Scrieți, în limbajul de programare studiat, un program care citește de la tastatură șase numere naturale, **g1**, **m1**, **s1**, respectiv **g2**, **m2**, **s2**, reprezentând măsurile (grade, minute și secunde, în această ordine) a două unghiuri ascuțite, adiacente. Programul afișează pe ecran măsura sumei celor două unghiuri, sub forma gradelor, minutelor și secundelor, în această ordine, separate prin câte un spațiu.

Exemplu: dacă se citesc numerele **30**, **40**, **56** și **82**, **35**, **24**, se afișează:

113 16 20 **(20p.)**

Barem de evaluare și de notare

Se acordă 10 puncte din oficiu. Punctajul maxim total este de 100 de puncte.

I.	**1.b 2.c**	**2 x 10p**	
II.			
1.	**pentru răspuns conform cerinței** -date constante conform cerinței (*) -date variabile conform cerinței (*)	**10p** 5p 5p	(*) Se acordă numai 2p dacă s-a precizat conform cerinței doar una dintre datele cerute, numai 3p dacă s-au precizat conform cerinței doar două dintre date, numai 4p dacă s-au precizat conform cerinței doar trei dintre date.
2.	**pentru răspuns conform cerinței** **A - 5; B - 3**	**10p**	Se acordă câte 5p pentru fiecare asociere conform cerinței.

3.	**pentru răspuns conform cerinței** -expresii în cadrul instrucțiunilor de decizie -cuvânt cheie conform cerinței	**10p** 2 x 3p 4p	
4.	**pentru răspuns conform cerinței** -valorile variabilelor precizate determinate conform cerinței pentru fiecare dintre cele **5** iterații -valoarea afișată	**20p** 5 x 3p 5p	
5.	**pentru răspuns conform cerinței** -citire date, conform cerinței -determinare a valorii secundelor din sumă -determinare a valorii minutelor din sumă (*) -determinare a valorii gradelor din sumă (*) -afișare date de ieșire, conform cerinței -corectitudine globală (sintaxă, alte aspecte neprecizate mai sus)	**20p** 2p 5p 5p 5p 2p 1p	(*) Se acordă numai 2p dacă la valoarea determinată nu s-a adăugat transportul.

Activități remediale recomandate

- În cazul unui răspuns necorespunzător cerinței I.1. se recomandă recapitularea operatorilor.
- În cazul unui răspuns necorespunzător cerinței I.2 se recomandă recapitularea structurilor de control liniară și repetitive.
- În cazul unui răspuns necorespunzător cerinței II.1 se recomandă discuție despre datele care se întâlnesc în textul unei probleme (de intrare, de ieșire, de manevră), valori constante și variabile.
- În cazul unui răspuns necorespunzător cerinței II.2 *Pentru A.* se va recapitula algoritmul pentru determinarea valorii maxime dintr-un șir de valori și a numărului de apariții a acesteia, se va urmări pas cu pas algoritmul pentru un set de date de intrare. *Pentru B.* se va utiliza metoda matematică de rezolvare a unui sistem de două ecuații cu două necunoscute, se va implementa metoda într-un algoritm informatic.
- În cazul unui răspuns necorespunzător cerinței II.3, dacă nu s-au completat corect punctele de suspensie care reprezintă expresia din cadrul structurii de decizie, se va recapitula algoritmul de determinare a valorii maxime dintr-un șir de valori citite, iar dacă nu s-a completat corect cuvântul cheie **`if`** se va recapitula structura de control și scrierea ei în limbajul de programare studiat.

- În cazul unui răspuns necorespunzător cerinței II.4 se va recapitula principiul de executare al structurilor repetitive și al structurii alternative. Se va urmări algoritmul pas cu pas pentru alte seturi de date de intrare și se va urmări evoluția valorilor variabilelor.
- În cazul unui răspuns necorespunzător cerinței II.5 se va exersa elaborarea unor algoritmi simpli.

2. B. 2. Metodă modernă/alternativă – probă practică

Specializarea matematică-informatică, intensiv informatică

Metoda utilizată:

"Turul Galeriei"

Timpul alocat pentru rezolvare:

45 de minute

Descrierea activității

1. Etape:

- Constituirea microgrupurilor: elevii sunt împărțiți în echipe formate din câte **4-5** membri; fiecare echipă își alege un nume și nominalizează unul dintre participanți ca ghid.
- Profesorul distribuie fiecărei echipe de elevi problema pe care trebuie să o soluționeze.

  Enunțuri posibile:
  - Se citește de la tastatură un număr natural din intervalul **[1,1000],** și se cere să se afișeze pe ecran media aritmetică a tuturor divizorilor săi proprii. Dacă numărul nu are divizori proprii se afișează pe ecran mesajul **Numarul nu are divizori proprii**.
  - Se citesc de la tastatură două numere naturale din intervalul **[2,20]**, **n** și **k,** apoi un șir de **n** numere naturale din intervalul **[10,1000]**. Se cere să se afișeze pe ecran numărul de valori divizibile cu **k** care sunt obținute din șirul dat prin ștergerea cifrei zecilor din fiecare termen al său.
  - Se citesc de la tastatură un număr natural **n** (**n∈[2,200]**), apoi un șir de **n** numere naturale din intervalul **[100,1000]**. Se cere să se afișeze pe ecran numărul din șir care are pe prima poziție cea mai mică cifră. Dacă există mai multe astfel de numere, se afișează cel mai mare dintre acestea.

- Șirul lui Fibonacci este definit astfel: $f_1=1, f_2=1, f_n=f_{n-1}+f_{n-2}$, pentru $n>2$.

  Se citesc de la tastatură două numere naturale **a** și **b** din intervalul $[1,40]$, $a \leq b$. Se cere să se afișeze pe ecran mesajul **Multiplu** dacă f_b este multiplu de f_a, unde atât f_a cât și f_b sunt termeni ai șirului lui Fibonacci, sau mesajul **Nu**, în caz contrar.
- Se citește de la tastatură un șir de numere naturale nenule din intervalul $[1,10^4]$, urmat de o valoare nulă, care nu face parte din șir. Se cere să se afișeze pe ecran perechea de valori citite consecutiv pentru care valoarea absolută a diferenței este maximă. Dacă există mai multe astfel de perechi, se va afișa ultima pereche citită.

- Profesorul precizează că soluția problemei trebuie inclusă într-un fișier (document sau prezentare), având numele echipei, într-un folder partajat de profesor cu întreaga clasă. Fișierul va avea următoarea structură:
  - Enunțul problemei primite spre rezolvare.
  - Numele echipei și numele și prenumele ghidului.
  - Numele și prenumele fiecărui membru, cerința/cerințele la care a contribuit la rezolvare, iar la fiecare cerință procentul în care a contribuit la realizarea acesteia.
  - O secțiune pentru soluțiile cerințelor.
  - O secțiune pentru Întrebări/Observații/Comentarii.
- Cooperarea pentru realizarea sarcinilor de lucru:
  - Elevii interacționează în cadrul microgrupurilor pentru a realiza sarcinile propuse.
  - Soluțiile se copiază în fișierul echipei, care este colaborativ.
- Expunerea produselor
  - Fiecare echipă copiază fișierul propriu în folderul clasei, care este colaborativ, acesta putând fi vizionat de ceilalți colegi, la fel ca într-o galerie de artă virtuală.
  - Elevii care au rolul de ghid vor urmări secțiunea Întrebări/Observații/Comentarii, răspunzând atunci când e cazul.
- Turul galeriei
  - Profesorul și membrii echipelor vizitează galeria, examinează fiecare produs, postează întrebări în vederea unor clarificări și/sau comentarii, pot completa ideile sau pot propune alte soluții pe care le consemnează în secțiunea corespunzătoare din fișierul echipei.
- Reexaminarea (evaluarea) rezultatelor
  - Fiecare grup își reexaminează propriile produse, prin comparație cu celelalte, valorificând comentariile vizitatorilor.
  - Profesorul evaluează elevii, putând apela și la interevaluare.

2. Avantaje ale metodei

- Stimularea creativității.
- Participarea activă, implicarea tuturor elevilor în realizarea sarcinilor de învățare.
- Dezvoltarea gândirii critice.
- Promovarea învățării active și a inter-învățării.
- Dezvoltarea competențelor de comunicare și de relaționare.
- Formarea și dezvoltarea competențelor de evaluare și autoevaluare.
- Cultivarea respectului față de ceilalți și a toleranței.
- Stimularea eforturilor de intercunoaștere și autocunoaștere.

3. Limite ale metodei:

- Aparentă dezordine.
- Neimplicarea unor elevi.
- Tendința de conformare la opinia grupului.
- Tendința unui/unor elevi de erijare în lideri, de dominare a grupului.
- Generarea unei gândiri de grup.

Cerințe:

1. **10 puncte**

Scrieți în fișierul echipei voastre datele de intrare și datele de ieșire identificate și tipurile de date utilizate în implementarea algoritmului în limbajul de programare studiat.

2. **10 puncte**

Scrieți în fișierul echipei voastre structurile de control utilizate în rezolvarea problemei.

3. **20 puncte**

Utilizând un mediu grafic interactiv, implementați algoritmul corespunzător enunțului primit. Inserați în fișierul echipei voastre o captură corespunzătoare rezolvării cerinței.

4. **20 puncte**

Implementați algoritmul utilizând un mediu de programare, folosind limbajele studiate de membrii grupului. Inserați în fișierul echipei codul sursă obținut.

5. **10 puncte**

Rulați programul pentru diverse seturi de date de intrare și inserați în fișierul echipei capturi cu datele de intrare și datele de ieșire corespunzătoare fiecărui caz posibil.

6. **10 puncte**

Vizionați soluțiile celorlalte echipe și scrieți în fișierele acestora întrebări/observații/comentarii pertinente.

7. **10 puncte**

Corectați eventualele erori, implementați modificările propuse dacă acestea corespund unor soluții eficiente, iar în caz contrar, argumentați opinia voastră.

Barem de evaluare și de notare

Se acordă 10 puncte din oficiu. Punctajul maxim total este de 100 de puncte.

1.	**pentru răspuns corect** -date de intrare (*) -date de ieșire (*) -tipuri corecte ale datelor de intrare (*) -tipuri corecte ale datelor de ieșire (*)	**10p.** 3p. 3p. 2p. 2p.	(*) Se acordă doar 1p dacă numai o parte dintre date/tipuri de date au fost precizate conform cerinței.
2.	**pentru răspuns corect** -structura secvențială -structura alternativă -structura repetitivă (*)	**10p** 3p. 3p. 4p.	(*) Se acordă numai 2p dacă se precizează structura repetitivă, fără tipul acesteia.
3.	**pentru implementare corectă** -declarare variabile -citire date -afișare date -blocuri de decizie (*) -blocuri repetitive (*) -atribuiri -corectitudine globală a implementării	**20p.** 2p. 2p. 2p. 5p. 5p. 2p. 2p.	(*) Se acordă numai 2p dacă blocurile sunt utilizate parțial corect.
4.	**pentru program corect** -declarare variabile -citire date -afișare date -instrucțiuni de decizie (*) -instrucțiuni repetitive (*) -atribuiri -corectitudine globală a programului	**20p.** 2p. 2p. 2p. 5p. 5p. 2p. 2p.	(*) Se acordă numai 2p dacă instrucțiunile sunt parțial corecte.

5.	**capturi pentru toate situațiile posibile** -capturi date de intrare (*) -capturi date de ieșire (*)	**10p.** 5p. 5p.	(*) Se acordă numai 3p dacă sunt tratate doar o parte dintre situațiile posibile.
6.	**vizionarea tuturor soluțiilor celorlalte echipe și postarea de întrebări/observații/comentarii pertinente** -vizualizarea produselor celorlalte echipe (*) -postarea de întrebări/ observații/ comentarii pertinente (**)	**10p.** 5p. 5p.	(*) Se acordă numai 3p dacă s-au vizionat produsele unui număr incomplet de echipe. (**) Se acordă numai 3p dacă s-au vizionat produsele unui număr incomplet de echipe sau dacă postările nu sunt la obiect.
7.	**Corectarea eventualelor erori, implementarea modificărilor propuse dacă acestea corespund unor soluții eficiente/argumentarea opiniilor echipei referitoare la propuneri** (*) -corectarea eventualelor erori -implementarea modificărilor propuse în vederea eficientizării/argumentarea opiniilor echipei referitoare la propuneri	**10p.** 5p. 5p.	(*) Se acordă 10p dacă nu au fost erori, respectiv nu au fost solicitate modificări/îmbunătățiri din partea profesorului sau a celorlalți colegi.

Activități remediale recomandate

- rezolvarea de probleme utilizând structura alternativă;
- rezolvarea de probleme utilizând structuri repetitive;
- implementarea unor algoritmi realizați anterior, în limbajul studiat.

2. C. Exemple de cerințe pentru realizarea evaluării inițiale

În această secțiune sunt propuse și alte exemple de itemi care pot fi integrate în teste inițiale sau utilizate pe parcursul recapitulării.

2. C. 1. Exemple de cerințe pentru evaluarea inițială – nivel 1 (cunoaștere)

2. C. 1. 1. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

1 (cunoaștere)

Enunț:

Identificați o situație care NU poate fi prelucrată cu ajutorul unui algoritm. Scrieți litera corespunzătoare răspunsului corect.

a. Știind disciplinele studiate la fiecare an de studiu, clasele dintr-o școală, profesorii și disciplinele pe care le pot preda aceștia, se cere orarul școlar.

b. Știind emisiunea preferată a fiecărui membru al unei familii de patru persoane și programul mai multor posturi TV, se cere numele persoanei care privește la unicul televizor al familiei, la ora curentă.

c. Știind datele elevilor dintr-o școală (numele și clasa), precum și notele obținute de aceștia, se cere numele celor care au premiul I.

d. Știind schița tehnică a unei mărci de automobile și având la dispoziție piesele componente corespunzătoare, se cere construcția unui astfel de automobil.

Răspuns corect: **b.**

2. C. 1. 2. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

1 (cunoaștere)

Enunț:

Scrieți două exemple de domenii ale activităţii umane în care este utilizat astăzi calculatorul.

Răspunsuri posibile:

- cercetare (de exemplu aplicaţii ale energiei nucleare, analize chimice şi medicale de mare precizie);
- procese tehnologice (de exemplu activităţi de proiectare, roboţi coordonaţi de calculator);
- activităţi cotidiene (de exemplu informaţii meteorologice, educaţie).

2. C. 1. 3. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

1 (cunoaștere)

Enunț:

În secvența alăturată de program, scrisă în limbajul C++, variabila **n** este de tip întreg. Analizați afirmațiile de mai jos și, pentru fiecare dintre acestea, scrieți litera **A** în cazul în care considerați că este adevărată, sau litera **F** în caz contrar.

```
do
{
     cout<<n%10;
     n=n/10;
}
while(n!=0);
```

1. Secvența respectă principiile programării structurate.
2. Secvența cuprinde o structură repetitivă cu test inițial.
3. Secvența cuprinde o atribuire.
4. Secvența cuprinde o operație de citire a datelor.

Răspuns corect: **1-A, 2-F, 3-A, 4-F.**

2. C. 2. Exemple de cerințe pentru evaluarea inițială – nivel 2 (aplicare)

2. C. 2. 1. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

2 (aplicare)

Enunț:

Se consideră secvența alăturată, scrisă în limbajul C++.

Stabiliți, prin săgeți, corespondența dintre cerințele din coloana stângă și valorile din coloana dreaptă.

```
cin>>x;
y=0;
while(x>0)
{
     y=y+x;
     x=x-2;
}
cout<<y;
```

A. dacă pentru **x** se citeşte valoarea **4**, în urma executării secvenței se afişează	**1.** 8
	2. 600
B. dacă pentru **x** se citeşte valoarea **50**, în urma executării secvenței se afişează	**3.** 6
	4. 650

Răspuns corect:

A-3 B-4

2. C. 2. 2. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

2 (aplicare)

Nivel cognitiv:

2 (aplicare)

Enunț:

În secvența alăturată, toate variabilele sunt de tip întreg.

```
cin>>n;
k=0;
 for(i=1;i<=n;i++)
 {
      cin>>x;
      if(x>=0 && x<10)
      {
           k=k+1;
      }
 }
 cout<<k;
```

a. Scrieți valoarea care se afișează în urma executării algoritmului dacă pentru **n** se citește valoarea **6**, iar pentru **x** valorile **12**, **15**, **1**, **5**, **9** și **22**.

b. Scrieţi **6** numere care pot fi citite pentru **x**, astfel încât, în urma executării algoritmului, pentru fiecare dintre acestea, valoarea afișată să fie **0**.

Răspunsuri corecte:

a. 3

b. Oricare **6** numere întregi care nu aparţin intervalului **[0, 9].**

2. C. 2. 3. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

2 (aplicare)

Enunț:

În secvența C/C++ alăturată, toate variabilele sunt întregi. De la tastatură se introduce un șir de **8** valori de câte cel mult **3** cifre, șir care conține cel puțin un număr par și cel puțin un număr impar. Scrieți o expresie care poate înlocui zona punctată astfel încât, în urma executării secvenței obținute, să se afișeze pe ecran valoarea celui mai mic număr par citit.

```
y=1000;
for(i=8;i>=1;i--)
{
      cin>>x;
      if(.....)
            y=x;
}
cout<<y;
```

Răspuns corect:

```
x%2==0 && x<y
```

Criterii de evaluare:

- identificare a unui număr par;
- identificare a unui număr mai mic decât y;
- condiție în ansamblu pentru actualizarea lui y.

2. C. 2. 4. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

2 (aplicare)

Enunț:

Pentru fiecare dintre cerințele de mai jos, precizați expresiile/cuvintele cheie cu care pot fi înlocuite punctele de suspensie din secvențele de program alăturate, scrise în limbajul C++, astfel încât secvența obținută să fie o soluție a cerinței respective.

a. Scrieți o secvență de program care să afișeze pe ecran primele **10** numere naturale impare.

```
i=1;
nr=0;
...(nr<10)
   {
      if(i%2==1)
      {
          cout<<i<<" ";
          nr++;
      }
      i=i+2;
   }
```

b. Scrieți o secvență de program care să afișeze consoanele din alfabetul limbii engleze, litere mici.

```
char x;
... (x='b';x<='z';x++)
{
 if(x!='e' && x!='i' && x!='o' && x!='u')
   cout<<x;
}
```

c. Scrieți o secvență de program care să citească de la tastatură un număr natural și să afișeze mesajul **numar prim**, dacă numărul citit este număr prim, **numar compus** dacă numărul citit este mai mare decât **2** și nu este prim, sau **nu este numar prim**, dacă numărul citit este mai mic decât **2**.

```
int n,i=2,gasit=0;
cin>>n;
if(n<=1)
  cout<<"nu este numar prim";
else
{
  if(n>2)
   do
   {
    if(n%i==0) gasit=1;
    else i=i+1;
   }
   ...(... && gasit==0);
  if(gasit==0)
      cout<<"numar prim";
  ... cout<<"numar compus";
 }
```

Răspuns corect:

a. `while`

b. `for`

c.
```
while(i*i<=n && gasit==0);
if(gasit==0)
     cout<<"numar prim";
else cout<<"numar compus";
```

2. C. 2. 5. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

2 (aplicare)

Enunț:

Azi e ziua codificărilor, fiecare număr este transformat într-un număr magic după secvența de program alăturată.

Scrieți litera **A**, dacă afirmația de mai jos este adevărată, sau litera **F** în caz contrar.

Dacă pentru **a** *se* citește *valoarea* **9299**, *atunci se afișează valoarea* **2**.

```
cin>>a;
if(a>9)
{ do
  { a=a%10+a/10; }
  while(a>9);
}
cout<<a;
```

Răspuns corect: **A.**

2. C. 3. Exemple de cerințe pentru evaluarea inițială – nivel 3 (raționament)

2. C. 3. 1. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

3 (raționament)

Enunț:

Scrieți în limbaj de programare o secvență de program care citește de la tastatură două numere întregi **x** (cu exact **5** cifre) și **y** (cu exact **6** cifre), și afișează pe ecran mesajul **DA**, dacă numărul format, în această ordine de la prima la ultima, din a **3**-a cifră a lui **x**, a **4**-a cifră a lui **y** și a **2**-a cifră a lui **y** este divizibil cu **7**, ca în exemplu, sau mesajul **NU** în caz contrar. Numerotarea cifrelor unui număr se face de la dreapta la stânga, începând cu **1** pentru cifra unităților.

Exemplu: dacă pentru **x** se citește valoarea **12345** și pentru **y** se citește valoarea **123467** răspunsul este **DA** (numărul format **336** este divizibil cu **7**).

Răspuns posibil:

```
cin>>x>>y;
if(((x/100%10)*100+(y/1000%10)*10+y/10%10)%7==0)
     cout<<"DA";
else
     cout<<"NU";
```

2. C. 3. 2. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

3 (raționament)

Enunț:

Pentru cerința de mai jos, identificați datele de intrare, datele de ieșire și datele de manevră.

Se consideră șase numere naturale **g1**, **m1**, **s1** și **g2**, **m2**, **s2**, care reprezintă măsurile a două unghiuri ascuțite adiacente, exprimate corespunzător în grade, minute, secunde. Se citesc valorile **g1**, **m1**, **s1** și **g2**, **m2**, **s2**, cu semnificația de mai sus, și se cere să se scrie pentru suma celor două unghiuri, în ordine, gradele, minutele și secundele, separate prin câte un spațiu.

Exemplu: dacă se citesc valorile **30 40 56** și **82 35 24**, se afișează pe ecran: **113 16 20**.

Criterii de evaluare și răspuns posibil:

- identificare a datelor de intrare: gradele, minutele și secundele celor două unghiuri (**g1**, **m1**, **s1** și **g2**, **m2**, **s2**);
- identificare a datelor de manevră: transportul spre minute, transportul spre grade; calcularea valorii gradelor din sumă;
- identificare a datelor de ieșire: numărul de grade, minute și secunde din sumă (**g m s**).

2. C. 3. 3. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

3 (raționament)

Enunț:

Andrei dorește să ajungă la un obiectiv turistic aflat la o distanță de **n** km. El parcurge **1** km în prima zi, apoi în fiecare zi care urmează, cu **1** km în plus față de ziua anterioară. Se cere numărul de zile în care Andrei parcurge drumul până la destinație.

Cerință:

Scrieți o secvență de program în limbajul C++ care citește de la tastatură un număr natural **n** și afișează numărul de zile în care Andrei ajunge la destinație.

Restricții și precizări:

10<=n<=1000000

În ultima zi, Andrei poate parcurge o distanță mai mică decât cea calculată, în cazul în care deja ajunge la destinație.

Exemplu:

Date de intrare	Date de ieșire	Explicație
14	**5**	În prima zi, Andrei parcurge **1** km; în a doua zi, **2** km; în a treia zi, **3** km; în a **4**-a zi, **4** km; în a **5**-a zi, **4** km (ar fi putut parcurge **5** km, dar se oprește după **4** km, deoarece a ajuns la destinație). Traseul va fi parcurs în **5** zile.

Criterii de evaluare și un posibil răspuns corect

-citire date

-afișare date

-determinare a numărului cerut (*)

-corectitudine globală a implementării

O soluție posibilă:

```
cin>>n;
nrzile=1;
total=1;
while(total<n)
{
   nrzile=nrzile+1;
   total=total+nrzile;
}
cout<<nrzile;
```

2. C. 3. 4. Exemplu de cerință pentru evaluarea inițială

Nivel cognitiv:

3 (raționament)

Enunț:

a. Scrieți instrucțiunile care pot înlocui punctele de suspensie, astfel încât în urma executării secvenței obținute, să se memoreze în variabila **sp** suma cifrelor pare ale lui **x**, iar în variabila **si** suma cifrelor impare ale lui **x**.

```
...
do....
{
  uc=.....;
  ....
}while(x!=0);
```

b. Rescrieți secvența alăturată astfel încât, în urma executării secvenței obținute, să se afișeze pe ecran toate numerele din intervalul [**a**,**b**] pentru care numărul **p** este număr par. Numărul **p** se obține scăzând din numărul **x** cifra unităților numărului **a** și adunând la rezultat cifra zecilor numărului **b**.

```
cin>>a>>b;
for(i=a;i<=b;i++)
{
  x=i;
  ....
}
```

Un posibil răspuns corect:

a.

```
sp=0;
si=0;
do
{
  uc=x%10;
  if(uc%2==0)
   sp=sp+uc;
  else
   si=si+uc;
  x=x/10;
}while(x!=0);
```

b.

```
cin>>a>>b;
for(i=a;i<=b;i++)
{
  x=i;
  p=x-a%10+b/10%10;

  if (p%2==0)
     cout<<x<<" ";
}
```

3. CONSTRUIREA NOILOR ACHIZIȚII

3. A. Recomandări pentru construirea noilor achiziții

Bazându-se pe rezultatele identificate în urma aplicării testelor inițiale, profesorul va proiecta activități de învățare adecvate, în vederea formării competențelor specifice corespunzătoare.

În această secțiune sunt prezentate exemple de activități de învățare pentru formarea/dezvoltarea competențelor specifice pentru specializările matematică-informatică, științe ale naturii și matematică-informatică intensiv informatică, cât și exemple pentru proiectarea unor activități de învățare. Profesorul va avea în vedere consolidarea achizițiilor din gimnaziu prin reluarea și abordarea unor elemente de conținut la un nivel corespunzător vârstei, în clasa a IX-a, acestea fiind reformulate în noi contexte de învățare, și va stabili modul de valorificare, completare și fuzionare a competențelor și conținuturilor din anii precedenți cu elementele componente din anul în curs.

Activitățile de învățare prezentate se pot utiliza în activitatea de predare-învățare-evaluare desfășurată atât în cadrul sălii de clasă, laborator, cât și online, context care a adus perspective noi învățării școlare. Astfel, tranziția de la gimnaziu la liceu poate deveni un proces eficient de învățare prin folosirea în continuare a resurselor digitale ca instrumente de facilitare a învățării și a comunicării între elevi și între elevi și profesori.

Setul de recomandări pentru construirea noilor achiziții cuprinde:

- pentru învățare se pot utiliza resurse didactice, bune practici, componente din curriculumul obligatoriu, sau platforme educaționale care facilitează transmiterea unor materiale didactice, de exemplu GSuite for Education cu aplicația Google Classroom, Moodle, Microsoft Teams, Edmodo, Learningapps;
- pentru teme, teste, exersare și feedback se pot utiliza platformele online dedicate disciplinei informatică, de exemplu pbinfo.ro, campion.edu.ro, varena.ro, infoarena.ro, precum și aplicațiile dedicate pentru reprezentarea testelor în format digital, de exemplu Google Forms, Microsoft Forms, Socrative, Go formative;
- pentru comunicare asincronă, alături de mijloacele integrate în platformele educaționale menționate mai sus, se pot utiliza aplicații specifice social media, ca instrumente utile pentru reflexivitate, bloguri care permit feedback (de exemplu Whatsapp, Facebook/Messenger, email, aplicații mobile);

- pentru comunicarea video online, alături de mijloacele integrate în platformele educaționale menționate mai sus, se pot utiliza aplicații dedicate (de exemplu Google Meet, Skype, Zoom, Microsoft Teams, Webex).

Activităţile de învățare pot fi proiectate și pentru grupe de elevi, având în vedere formarea competenţelor din programa şcolară şi în plus dezvoltarea competenţelor esenţiale pentru profesiile secolului XXI:

- responsabilitate: elevul îşi îndeplineşte sarcinile de lucru şi le predă în formatul solicitat, respectând timpul limită de predare; elevul îşi asumă responsabilitatea pentru propriul comportament;

- organizare: elevul realizează un plan în care detaliază sarcinile pe care trebuie să le îndeplinească pentru a finaliza cu succes activitatea; stabileşte priorităţi; gestionează eficient timpul de lucru; evaluează obiectiv modul în care şi-a îndeplinit sarcinile de lucru; alege resursele informaţionale şi tehnologice adecvate pentru îndeplinirea sarcinilor de lucru;

- independenţă în învăţare: elevul este capabil să îşi planifice, monitorizeze, evalueze, gestioneze propriul proces de învăţare, pentru a atinge obiectivele; demonstrează că are iniţiativă, curiozitate şi interes pentru învăţare; demonstrează creativitate în modul de abordare a unor probleme noi;

- colaborare: elevul este capabil să lucreze în echipă, asumându-şi diferite roluri în cadrul echipei, realizându-şi propriile sarcini de lucru, contribuind cu idei, opinii, soluţii la buna desfăşurare a întregului proiect, comunicând constructiv cu colegii de echipă, partajând resurse sau expertiza proprie, contribuind la rezolvarea conflictelor şi la luarea deciziilor.

Limbajul de programare studiat în clasa a IX-a va fi ales conform programei școlare în vigoare pentru clasa a IX-a (C, C++ sau Pascal). Având în vedere că mediile de programare pentru editarea, depanarea și rularea programelor în limbajul Pascal nu sunt actualizate și nu sunt compatibile cu majoritatea sistemele de operare actuale, **se recomandă utilizarea limbajului C++**, având în vedere și oportunitățile pe care le oferă privind programarea orientată pe obiecte, studiată în clasele mai mari.

În funcție de particularitățile clasei de elevi, se pot întâlni situații în care o parte dintre elevii clasei să fi studiat un limbaj de programare în ciclul gimnazial, iar o altă parte dintre elevi să fi studiat un alt limbaj de programare. Unii elevi vor avea deci nevoie de o trecere spre un alt limbaj de programare, studiat la liceu. Pentru ca acest proces să fie treptat, recomandăm ca, pentru început, descrierea algoritmilor în limbaj de programare să se facă în paralel în limbajele studiate de elevii

clasei, astfel încât aceștia să poată observa similitudini și particularități ale noului limbaj de programare studiat, în comparație cu limbajul studiat în ciclul gimnazial. Profesorul va accentua și evidenția diferențele și asemănările între programele scrise în aceste limbajele de programare.

După câteva săptămâni de acomodare a elevilor cu limbajul de programare ales, conform programei în vigoare, algoritmii vor putea fi implementați doar în acest limbaj. La clasele cu specializarea matematică-informatică sau științe ale naturii se recomandă elaborarea programelor în pseudocod, dar și implementarea programelor în limbajul de programare, pentru valorificarea competențelor formate în gimnaziu și pentru asigurarea continuității învățării.

Pentru a forma competențele specifice precizate în programa școlară, profesorii au posibilitatea să adapteze permanent metodele didactice la nivelul clasei. Materialul conține mai multe exemple de activități de învățare, sugestii metodologice, menite să sprijine cadrele didactice în alegerea unor scenarii didactice eficiente, a unor activități interdisciplinare, realizând astfel conexiuni cu discipline ca matematică, fizică, chimie și orice domenii în care se pot face statistici, se prelucrează date în mod repetitiv sau se prelucrează volume mari de date într-un timp scurt, cu ajutorul informaticii.

3. B. Exemple de activități de învățare pentru formarea/dezvoltarea competențelor specifice

În cele ce urmează sunt prezentate, pentru fiecare competență specifică, exemple de activități de învățare pentru activitatea didactică de pe parcursul clasei a IX-a.

3. B. 1. Exemple de activități de învățare - Specializările matematică-informatică, științe ale naturii

1. Identificarea conexiunilor dintre informatică şi societate

1.1. Identificarea aplicaţiilor informaticii în viaţa socială

- Identificarea domeniilor de aplicabilitate a informaticii, de exemplu: viața cotidiană, educație, domeniul de cercetare, sistemul medical, turism, procese tehnologice din industrie.
- Discutarea efectelor pozitive și negative ale utilizării aplicațiilor informatice și înregistrarea rezultatului discuțiilor într-un document colaborativ la care să aibă acces toți elevii.
- Identificarea de echipamente electronice utilizate în viața de zi cu zi, care sunt programabile, de exemplu: roboții, dispozitivul GPS.

1.2. Recunoaşterea situaţiilor în care este necesară prelucrarea algoritmică a informaţiilor

- Prezentarea unor activități cotidiene care se desfășoară prin parcurgerea unei secvențe de pași, de exemplu: realizarea unui fel de mâncare urmând pașii descriși într-o rețetă culinară și folosind ingredientele necesare, construirea unei case, rezervarea online a unui bilet la un spectacol, comandarea online a unui produs.
- Prezentarea unor algoritmi cunoscuți care au fost studiați la disciplina matematică și care necesită parcurgerea unei secvențe de pași pentru rezolvarea cerințelor, de exemplu: rezolvarea ecuației de gradul al doilea, algoritmul lui Euclid pentru determinarea celui mai mare divizor comun a două numere naturale.
- Identificarea unor situații în care nu se poate aplica un algoritm, de exemplu: determinarea tuturor multiplilor unui număr natural dat, determinarea emisiunilor preferate ale membrilor unei familii.
- Simularea unor fenomene naturale sau procese tehnologice în vederea studierii efectelor acestora (fenomene meteo, mișcarea astrelor, combustie în cuptoare industriale).
- Analiza modului în care sunt realizate sondaje, studii, cercetări pornind de la colectarea datelor până la publicarea rezultatelor acestora.

2. Identificarea datelor care intervin într-o problemă şi a relaţiilor dintre acestea

2.1. Descompunerea rezolvării unei probleme în paşi

- Analiza unei activități cotidiene care presupune o secvență de pași în realizarea ei, de exemplu: apelarea unei persoane cunoscute cu ajutorul telefonului mobil și identificarea datelor necesare pentru rezolvare, traseul unui robot pe o planșă de lucru, astfel încât să nu atingă marginile acesteia, identificând datele necesare pentru rezolvare.
- Analiza unei probleme cunoscute de la disciplinele matematică sau fizică și identificarea etapelor de rezolvare, de exemplu: determinarea necunoscutei unei ecuații de gradul I în care se cunosc coeficienții, înregistrarea temperaturii pe parcursul unei săptămâni și calcularea temperaturii medii din săptămâna respectivă.

2.2. Identificarea tipurilor de date necesare pentru rezolvarea unei probleme: de intrare, de ieşire, de manevră

- Explicarea noțiunilor de dată și informație printr-un demers inductiv care parcurge etapele descoperirii de la particular (exemple care conțin situațiile ce urmează a fi identificate) la general (formularea definițiilor), cu revenire la particular pentru fixare (prin exerciții de: recunoaștere, caracterizare, motivare, disociere, exemplificare).

- Identificarea unor modalități de transmitere a unor date/informații între membrii unei comunități și identificarea datelor/informației sursă și a datelor/informației finale.
- Identificarea unor modalități de extragere a datelor relevante pentru stocarea informației în vederea utilizării ulterioare.
- Identificarea datelor de intrare, de ieșire, de manevră pentru un exercițiu matematic care admite o prelucrare algoritmică (de exemplu: pentru două mulțimi A și B pentru care se cunosc valorile, numere naturale, se vor determina datele de intrare și datele de ieșire rezultate din operațiile de reuniune, intersecție și produs cartezian, utilizând metodele matematice studiate în gimnaziu).

2.3. Descrierea coerentă a unei succesiuni de operaţii prin care se obţin din datele de intrare, datele de ieşire

- Urmărirea pas cu pas a unei secvențe dintr-un algoritm pentru diferite seturi de date de intrare, înregistrarea rezultatelor obținute și descrierea în limbaj natural a prelucrării realizate.
- Identificarea operatorilor și datelor de intrare, de manevră și de ieșire dintr-o secvență algoritmică, precum și a etapelor/pașilor prin care acestea își modifică valorile, în vederea obținerii datelor de ieșire.

3. Elaborarea algoritmilor de rezolvare a problemelor

3.1. Analizarea enunţului unei probleme: identificarea datelor de intrare, a datelor de ieşire cu specificarea tipului datelor, a relaţiilor existente între date şi stabilirea paşilor de rezolvare a problemei

- Descrierea, pas cu pas, a rezolvării unei probleme și divizarea ei în etape/probleme mai simple care, la rândul lor, sunt analog divizate.
- Scrierea unui algoritm pentru rezolvarea unei probleme date care utilizează, în mod repetat, o secvență de prelucrare, în vederea identificării etapelor independente, precum și a șabloanelor, caracteristicilor repetitive, de exemplu: determinarea celui mai mare divizor comun a două numere naturale, determinarea sumei cifrelor unui număr natural care are cel puțin 3 cifre.

3.2. Reprezentarea algoritmilor în pseudocod

- Argumentarea utilizării pseudocodului ca o formă de reprezentare a unui algoritm independentă de tehnologia/limbajul de programare utilizat, ca punct de pornire pentru discuții în cadrul unui proiect complex.

- Identificarea cuvintelor cheie întâlnite într-un algoritm liniar.
- Scrierea în pseudocod a structurilor de control liniare prin analogie cu blocurile grafice și valorificând limbajul de programare studiat în gimnaziu.
- Scrierea în pseudocod a structurilor de control de decizie prin analogie cu blocurile grafice și valorificând limbajul de programare studiat în gimnaziu.
- Scrierea în pseudocod a structurilor de control repetitive și evidențierea asemănărilor și a diferențelor între acestea: cu număr cunoscut de pași, cu test final şi cu test inițial, prin analogie cu blocurile grafice și valorificând limbajul de programare studiat în gimnaziu.
- Scrierea unor algoritmi în limbajul pseudocod care conțin operațiile de citire, de scriere, de atribuire și de decizie și identificarea cazurilor particulare, de exemplu: media aritmetică a trei numere, maximul a două numere, ultima cifră a unui număr, scrierea unei expresii, prin analogie cu blocurile grafice și valorificând limbajul de programare studiat în gimnaziu.
- Construirea unor algoritmi pentru: prelucrarea numerelor (la nivel de cifră, divizibilitate), precum şi a secvenţelor de numere (determinarea elementului minim/maxim, verificarea unor proprietăți, generarea unor şiruri recurente).
- Urmărirea pas cu pas a unor secvenţe de algoritm, cu urmărirea evoluției variabilelor și analizând efectele executării structurilor de control.
- Adaptarea unor algoritmi existenți în scopul rezolvării unor probleme asemănătoare.
- Înlocuirea diferitelor tipuri de structuri repetitive și de structuri de decizie, prin construirea unui algoritm echivalent cu un algoritm dat.

3.3. Respectarea principiilor programării structurate în procesul de elaborare a algoritmilor

- Analiza unor algoritmi construiți și identificarea tipurilor de structuri de control utilizate. Modificarea algoritmilor prin înlocuirea unei structuri de control cu o altă structură echivalentă, evidențiindu-se reprezentarea oricărei prelucrări cu ajutorul structurilor de control studiate.
- Scrierea în pseudocod a unor algoritmi elementari și urmărirea pas cu pas a execuției acestora, evidențiind unele cazuri particulare tratate corespunzător și siguranța în funcționarea algoritmului, de exemplu: prelucrare a cifrelor unui număr, divizibilitate, prelucrare a unor secvențe de valori).
- Construirea de algoritmi obținuți prin compunerea sau combinarea unor algoritmi elementari cunoscuți, evidențiindu-se reprezentarea oricărei prelucrări cu ajutorul structurilor de control studiate.

- Construirea unor algoritmi care tratează o anumită temă, aplicând pe rând algoritmii de bază pentru prelucrarea unei serii de valori, evidențiind secvențele care se păstrează de la o prelucrare la alta și structurile de control utilizate, de exemplu: afișarea cifrelor unui număr pe rânduri diferite, numărarea cifrelor unui număr, suma cifrelor unui număr, cifra maximă a unui număr.

4. Aplicarea algoritmilor fundamentali în prelucrarea datelor

4.1. Elaborarea unui algoritm de rezolvare a unor probleme din aria curriculară a specializării

- Scrierea în pseudocod a algoritmilor de rezolvare a unei probleme din aria curriculară a specializării, valorificând limbajul de programare studiat, de exemplu: implementarea algoritmilor pentru rezolvarea de aplicaţii interdisciplinare, cum ar fi: rezolvarea ecuaţiei de gradul I şi de gradul al II-lea, simplificarea fracţiilor, calcularea sumei/produsului unui șir de fracții și scrierea rezultatului sub formă de fracție ireductibilă, generarea primilor **n** termeni ai unei progresii aritmetice cu primul termen și rația cunoscute, aplicaţii geometrice (distanţa dintre două puncte, aria/perimetrul unui triunghi, volumul corpurilor regulate), determinarea punctului de intersecţie a două mobile în mişcare rectilinie şi uniformă, determinarea masei moleculare a unui compus chimic.
- Exersarea executării algoritmilor pentru rezolvarea unor probleme întâlnite de elevi în studiul altor discipline şcolare sau în viaţa cotidiană evidențiind posibilitatea obținerii rapide a rezultatelor pentru o clasă de probleme de același tip.
- Exersarea testării și depanării programelor, evidențiind importanța alegerii datelor de intrare relevante pentru comportamentul programului și pentru problema propriu-zisă pe care o rezolvă.

4.2. Alegerea unui algoritm eficient de rezolvare a unei probleme

- Exersarea și simularea unor algoritmi echivalenți pentru rezolvarea unor probleme și calcularea numărului de operații pe care le execută fiecare algoritm pentru seturi de date de intrare, de exemplu: calcularea sumei primilor **n** termeni ai unei progresii aritmetice cu primul termen și rația cunoscute, prin utilizarea structurilor de control repetitive și prin utilizarea formulelor matematice.
- Compararea algoritmilor cunoscuți pentru determinarea celui mai mare divizor comun din punctul de vedere al timpului de executare și calcularea numărului de iterații pe care le execută fiecare algoritm pentru aceleași date de intrare.

- Compararea algoritmilor cunoscuți pentru identificarea numărului de divizori ai unui număr natural **n**, din punctul de vedere al timpului de executare, de exemplu prin parcurgerea intervalului **[1,n]**, **[2,n/2]** sau **[1,√n]**.
- Considerații privind eficiența unui algoritm din punctul de vedere al timpului de executare prin calcularea numărului de operații efectuate.
- Testarea şi analizarea comportamentului programelor pentru diferite date de intrare din perspectiva numărului de operații executate, respectiv a timpului de executare.
- Discuţii şi dezbateri asupra diferiților algoritmi echivalenți care se pot scrie pentru o problemă dată, cu evidenţierea celor optimi din punctul de vedere al numărului de operații/timpului de executare.

5. Implementarea algoritmilor în limbaj de programare

5.1. Identificarea într-un program a structurilor de control învăţate

- Identificarea structurii secvențiale și a structurii alternative cu implementarea în limbajul C++ a algoritmilor fundamentali pentru numărarea unor valori, calculul unor sume, produse, minime, maxime, verificare a unor proprietăți.
- Identificarea structurii repetitive cu test inițial și a structurii repetitive cu număr cunoscut de pași cu implementarea în limbaj C++ și combinații ale acestora, de exemplu: algoritmul pentru calcularea sumei unui șir de fracții și scrierea rezultatului sub formă de fracție ireductibilă.
- Identificarea structurii repetitive cu test final cu implementarea în limbaj C++, de exemplu: algoritmul pentru determinarea numărului de cifre ale unui număr natural, în care este utilizată structura repetitivă de control cu test final, prin evidențierea unor rezultate greșite în cazul utilizării structurii repetitive cu test inițial.
- Finalizarea unui produs informatic care presupune funcţionarea acestuia conform condiţiilor impuse de beneficiar, însoţirea lui de o documentaţie, verificarea, testarea şi evaluarea sa.

3. B. 2. Exemple de activități de învățare - Specializarea matematică-informatică, intensiv informatică

1. Identificarea conexiunilor dintre informatică şi societate

1.1. Identificarea aplicaţiilor informaticii în viaţa socială

- Identificarea domeniilor de aplicabilitate a informaticii, de exemplu: viața cotidiană, educație, domeniul de cercetare, sistemul medical, turism, procese tehnologice din industrie.
- Discutarea efectelor pozitive și negative ale utilizării aplicațiilor informatice și înregistrarea rezultatului discuțiilor într-un document colaborativ la care să aibă acces toți elevii.
- Identificarea de echipamente electronice utilizate în viața de zi cu zi, care sunt programabile, de exemplu: roboții, dispozitivul GPS.

1.2. Recunoaşterea situaţiilor în care este necesară prelucrarea algoritmică a informaţiilor

- Prezentarea unor activități cotidiene care se desfășoară prin parcurgerea unei secvențe de pași, de exemplu: realizarea unui fel de mâncare urmând pașii descriși într-o rețetă culinară și folosind ingredientele necesare, construirea unei case, rezervarea online a unui bilet la un spectacol, comandarea online a unui produs.
- Prezentarea unor algoritmi cunoscuți care au fost studiați la disciplina matematică și care necesită parcurgerea unei secvențe de pași pentru rezolvarea cerințelor, de exemplu: rezolvarea ecuației de gradul al doilea, algoritmul lui Euclid pentru determinarea celui mai mare divizor comun a două numere naturale.
- Identificarea unor situații în care nu se poate aplica un algoritm, de exemplu: determinarea tuturor multiplilor unui număr natural dat, determinarea emisiunilor preferate ale membrilor unei familii.
- Simularea unor fenomene naturale sau procese tehnologice în vederea studierii efectelor acestora (fenomene meteo, mișcarea astrelor, combustie în cuptoare industriale).
- Analiza modului în care sunt realizate sondaje, studii, cercetări pornind de la colectarea datelor până la publicarea rezultatelor acestora.

2. Identificarea datelor care intervin într-o problemă şi a relaţiilor dintre acestea

2.1. Descrierea unei succesiuni de operaţii prin care se obţin din datele de intrare, datele de ieşire

- Descrierea unui algoritm în limbaj natural, identificând datele necesare pentru rezolvare și care nu pot fi obținute prin prelucrare, rezultatele și datele obținute pe parcursul prelucrării.
- Prezentarea etapelor rezolvării unei probleme cu ajutorul unui algoritm informatic și identificarea acestora în cadrul unor exemple.
- Identificarea unor algoritmi specifici disciplinelor studiate în gimnaziu, respectiv a unor probleme a căror rezolvare presupune descompunerea în pași, de exemplu: algoritmul lui Euclid pentru determinarea celui mai mare divizor comun a două numere naturale.

3. Elaborarea algoritmilor de rezolvare a problemelor

3.1. Analizarea enunţului unei probleme şi stabilirea paşilor de rezolvare a problemei

- Urmărirea pas cu pas a unei secvențe dintr-un algoritm pentru diferite seturi de date de intrare, înregistrarea rezultatelor obținute și descrierea în limbaj natural a prelucrării realizate.
- Identificarea operatorilor și a datelor de intrare, de manevră și de ieșire dintr-o secvență algoritmică, precum și a etapelor/pașilor prin care acestea își modifică valorile, în vederea obținerii datelor de ieșire.

3.2. Reprezentarea algoritmilor în pseudocod

- Argumentarea utilizării pseudocodului ca o formă de reprezentare a unui algoritm independentă de tehnologia/limbajul de programare utilizat, ca schiță/punct de pornire care cuprinde o descriere clară a unor etape în cadrul unui proiect complex.
- Exersarea scrierii unor algoritmi secvențiali prin analogie cu blocurile grafice și valorificând limbajul de programare studiat în gimnaziu, de exemplu: determinarea valorilor unor expresii pentru anumite date de intrare, determinarea ariilor și volumelor unor corpuri geometrice studiate.
- Exersarea scrierii unor algoritmi utilizând structura alternativă simplă, respectiv structura alternativă multiplă prin analogie cu blocurile grafice și valorificând limbajul de programare studiat în gimnaziu, de exemplu: rezolvarea ecuației de gradul I și a ecuației de gradul al II-lea, determinarea intersecției a două intervale, determinarea minimului și maximului a unui număr mic de valori.

- Identificarea situațiilor în care utilizăm structurile repetitive și scrierea algoritmilor corespunzători și evidențierea asemănărilor și a diferențelor între acestea: cu număr cunoscut de pași, cu test final și cu test inițial prin analogie și valorificând blocurile grafice limbajul de programare studiate la gimnaziu, de exemplu: determinarea sumei, produsului, numărului de valori cu o anumită proprietate, pentru **n** valori citite, respectiv, pentru valori citite până la îndeplinirea unei condiții.

3.3. Respectarea principiilor programării structurate în procesul de elaborare a algoritmilor;

- Prezentarea teoremei programării structurate a lui Bohm și Jacopini și recunoașterea structurilor fundamentale utilizate în cadrul unor algoritmi dați.
- Exersarea scrierii de algoritmi structurați și urmărirea pas cu pas a execuției acestora, evidențiind unele cazuri particulare tratate corespunzător și siguranța în funcționarea algoritmului, de exemplu: probleme care prelucrează cifrele unor numere, probleme de divizibilitate, probleme de numărare, probleme de determinare a valorilor maxime/minime, determinarea unor situații statistice.
- Construirea de algoritmi obținuți prin combinarea unor algoritmi elementari cunoscuți, evidențiindu-se reprezentarea oricărei prelucrări cu ajutorul structurilor de control studiate.
- Construirea unor algoritmi care tratează o anumită temă, aplicând pe rând algoritmii de bază pentru prelucrarea unei serii de valori, evidențiind secvențele care se păstrează de la o prelucrare la alta și structurile de control utilizate, de exemplu: afișarea cifrelor unui număr pe rânduri diferite, numărarea cifrelor unui număr, suma cifrelor unui număr, cifra maximă a unui număr.
- Prezentarea unor exemple de algoritmi nestructurați.
- Testarea şi analizarea comportamentului algoritmilor pentru diferite date de intrare.
- Evidenţierea greşelilor tipice în elaborarea algoritmilor, de exemplu inițializarea datelor la nivel global/în cadrul unor structuri repetitive, omiterea unor cazuri particulare.

4. Implementarea algoritmilor într-un limbaj de programare

4.1. Transcrierea algoritmilor din pseudocod într-un limbaj de programare

- Prezentarea unor exemple de implementare în mai multe limbaje de programare a unor algoritmi elaborați și executarea acestora pe calculator, evidențiind prezența unor elemente reper în sintaxa instrucțiunilor de bază ale fiecăruia dintre acestea, precum și diferențe ale scrierii acestor instrucțiuni.

- Prezentarea mediului de programare, facilităţi de editare, de compilare şi de rulare. Exersarea testării și depanării programelor, evidențiind importanța alegerii datelor de intrare relevante pentru comportamentul programului și pentru problema propriu-zisă pe care o rezolvă.
- Exersarea citirii datelor de la tastatură şi a afişării datelor pe ecran.
- Activități de scriere în limbajul de programare C++ a structurilor de control studiate.

4.2. Identificarea necesităţii structurării datelor în tablouri

- Exemplificarea unor situații în care rezultatul nu se poate obține pe măsura citirii datelor și identificarea necesității structurării datelor în tablouri unidimensionale și bidimensionale, de exemplu: determinarea valorii maxime dintr-un șir de numere, precum și pozițiile pe care apare, păstrarea unor valori care sunt în relație cu una sau două alte valori.
- Analizarea unor exemple din viața cotidiană în care sunt necesare prelucrări ale datelor structurate, de exemplu: bugetul de venituri și cheltuieli al familiei, date statistice referitoare la rezultatele obținute de elevii unei clase la examenele naționale, înregistrarea și analizarea măsurătorilor specifice unui experiment efectuat în laboratorul de fizică, analizarea valorilor unei funcții într-un interval de valori.
- Reprezentarea tablourilor unidimensionale în memorie și efectuarea operațiilor specifice cu datele memorate în acestea, evidențiind necesitatea/facilitățile utilizării unei astfel de structuri.
- Reprezentarea tablourilor bidimensionale în memorie și efectuarea operațiilor specifice cu datele memorate în acestea, evidențiind necesitatea/facilitățile utilizării unei astfel de structuri.

4.3. Prelucrarea datelor structurate

- Analizarea unei probleme simple în scopul identificării necesității utilizării datelor structurate și evidențierea operațiilor de prelucrare a acestora în funcție de tipul de date al elementelor memorate.

Tablouri unidimensionale

- Construirea unor algoritmi și implementarea programelor în limbajul de programare C++ pentru aplicații care presupun parcurgerea elementelor tablourilor unidimensionale în ordinea memorării, în ordine inversă a memorării, accesul direct la un element aflat pe o anumită poziție, și a aplicării algoritmilor de bază pentru prelucrarea seriilor de valori (numărare, însumare, calcul de produse, minime/maxime, verificare de proprietăți).

- Construirea unor algoritmi și implementarea programelor în limbajul de programare C++ pentru aplicații care presupun inserări și ștergeri de valori memorate în tablourile unidimensionale.
- Construirea unor algoritmi și implementarea programelor în limbajul de programare C++ pentru aplicații care presupun sortarea elementelor din tablourile unidimensionale utilizând mai multe metode de sortare: metoda selecției, metoda bulelor, metoda numărării, metoda inserției, metoda interschimbării, evidențiind particularitățile acestora.
- Construirea unor algoritmi și implementarea programelor în limbajul de programare C++ pentru aplicații care presupun interclasări între elementele tablourilor unidimensionale.
- Construirea unor algoritmi și implementarea programelor în limbajul de programare C++ pentru aplicații care presupun căutări secvențiale și binare în tablourile unidimensionale.
- Construirea unor algoritmi și implementarea programelor în limbajul de programare C++ pentru aplicații care presupun lucrul cu secvențe în tablourile unidimensionale.
- Construirea unor algoritmi și implementarea programelor în limbajul de programare C++ pentru aplicații care presupun utilizarea vectorilor caracteristici/de frecvență.

Tablouri bidimensionale

- Construirea unor algoritmi și implementarea programelor în limbajul de programare C++ pentru aplicații care presupun parcurgerea elementelor tablourilor bidimensionale pe linii/pe coloane, accesul direct la un element aflat pe o anumită poziție, aplicarea raționamentelor specifice tablourilor unidimensionale pentru prelucrarea datelor aflate pe o linie/coloană a tabloului.
- Construirea unor algoritmi și implementarea programelor în limbajul de programare C++ pentru aplicații care presupun parcurgerea și prelucrarea elementelor din tablourile bidimensionale pătratice, de exemplu: identificarea zonelor din tablou în funcție de diagonala principală și secundară.
- Construirea unor algoritmi și implementarea programelor în limbajul de programare C++ pentru aplicații care presupun parcurgerea elementelor tabloului cu evidențierea facilităților aduse de bordarea acestuia, de exemplu identificarea elementelor care au toți vecinii nenuli.
- Construirea unor algoritmi și implementarea programelor în limbajul de programare C++ pentru aplicații care presupun generări de matrice ce au anumite proprietăți deduse pe baza unei formule de calcul în funcție de poziția lor în matrice și reprezentarea acestora ca tablouri bidimensionale, de exemplu: matricea memorează șirul valorilor pare, fiecare

element memorează valoarea maximă determinată dintre indicele de linie și indicele de coloană, fiecare element este egal cu produsul indicilor săi.

- Urmărirea pas cu pas a algoritmilor construiţi și a programelor realizate pentru diferite seturi de date de intrare, cu identificarea eventualelor cazuri speciale.

4.4. Utilizarea fişierelor text pentru introducerea datelor şi extragerea rezultatelor

- Identificarea datelor de intrare și ieșire pentru o problemă familiară, din viața cotidiană, întâlnită la una dintre disciplinele studiate, și evidențierea facilităților aduse de memorarea acestora într-un fișier.
- Identificarea necesității memorării datelor de intrare sau de ieșire într-un fișier text, de exemplu: parcurgerea unor date de intrare sau de ieșire sub forma unui șir de `1000000` de valori întregi.
- Descrierea și utilizarea operațiilor specifice utilizării fișierelor de intrare/ieșire.
- Evidenţierea analogiilor şi diferenţelor între citire/scriere utilizând dispozitivele standard de intrare/ieşire şi utilizarea fişierelor text.

4.5. Utilizarea unui mediu de programare pentru limbajul C++

- Realizarea codului sursă prin transcrierea structurilor algoritmului cu ajutorul instrucțiunilor/comenzilor specifice limbajului, evidențiind facilitățile mediului de programare pentru alinierea instrucțiunilor, relaționarea unor părți ale instrucțiunilor (if-else, do-while, acolade) semnalarea unor erori de sintaxă pe măsura scrierii codului.
- Rularea linie cu linie sau prin salt la o anumită linie a unei surse existente pentru urmărirea valorilor variabilelor din program, respectiv pentru identificarea erorilor de implementare sau de proiectare a algoritmului în vederea depanării codului sursă.
- Evidențierea timpului de executare a unor programe care rezolvă aceeași problemă, având o eficiență diferită.
- Realizarea unor aplicații interdisciplinare elementare, de exemplu: realizarea graficului precipitațiilor într-un interval de timp dat, determinarea defrișărilor pe o anumită suprafață de formă dreptunghiulară, analiza cursului valutar pe parcursul celor `12` luni ale unui an calendaristic, calculul frecvenței temperaturilor minime și maxime înregistrate în ultimii `50` de ani, modelarea relațiilor de prietenie între colegii unei clase de elevi.

5. Aplicarea algoritmilor fundamentali în prelucrarea datelor

5.1. Elaborarea unui algoritm de rezolvare a unor probleme din aria curriculară a specializării

- Identificarea unor aplicații la disciplina matematică și realizarea algoritmilor de rezolvare precum și implementarea programelor corespunzătoare în limbaj de programare, de exemplu: calculul termenilor unei progresii aritmetice, calculul suprafețelor diferitelor figuri din geometria plană, rezolvarea ecuațiilor liniare, rezolvarea unui sistem de două ecuații cu două necunoscute, poziția relativă a două cercuri în plan, determinarea intersecției mai multor intervale de valori.
- Identificarea unor aplicații la disciplina fizică și realizarea algoritmilor de rezolvare precum și implementarea programelor corespunzătoare în limbaj de programare, de exemplu: calculul vitezei de deplasare a unui mobil între două repere date, calculul unghiului de refracție al unui fascicul de lumină.
- Identificarea unor aplicații la disciplina chimie și realizarea algoritmilor de rezolvare precum și implementarea programelor corespunzătoare în limbaj de programare, de exemplu: calculul masei moleculare a unei substanțe chimice, determinarea numărului de izomeri al unui compus chimic, determinarea acidității unei probe de apă, identificarea proprietăților fizice ale unui element în funcție de poziția acestuia în tabelul periodic al lui Mendeleev.

5.2. Alegerea unui algoritm eficient de rezolvare a unei probleme

- Considerații privind eficiența unui algoritm din punctul de vedere al timpului de executare prin calcularea numărului de operații elementare efectuate.
- Exersarea și simularea unor algoritmi echivalenți pentru rezolvarea unor probleme și calcularea numărului de operații pe care le execută fiecare algoritm pentru seturi de date de intrare, de exemplu: calcularea sumei primilor **n** termeni ai unei progresii aritmetice, cu primul termen și rația cunoscute, prin utilizarea structurilor de control repetitive și prin utilizarea formulelor matematice.
- Analizarea unor algoritmi ce rezolvă aceeași problemă, în funcție de eficiența lor și selectarea celui mai eficient algoritm, de exemplu: compararea algoritmilor cunoscuți pentru identificarea numărului de divizori ai unui număr natural **n**, din punctul de vedere al timpului de executare, de exemplu prin parcurgerea intervalului `[1,n]`, `[2,n/2]` sau `[1,√n]`.
- Optimizarea unui algoritm din punctul de vedere al timpului de executare.
- Optimizarea unui algoritm din punctul de vedere al memoriei utilizate.

3. C. Proiectarea unei activități de învățare - exemple

Proiectarea demersului didactic este un proces prin care cadrul didactic stabilește atât etape cât și acțiuni de parcurs și de realizat în procesul de predare-învățare-evaluare.

Activitatea de învățare reprezintă cadrul de formare a unei competențe specifice, având în vedere o modalitate de organizare a activității didactice în acest scop, iar activitatea de învățare presupune implicarea elevului într-un ansamblu de sarcini de lucru, în mod direct relevante pentru formarea sau dezvoltarea unei competențe specifice.

O modalitate propusă pentru proiectarea unei activități de învățare cere stabilirea unor repere ca:

- titlul activității de învățare: o secvență de text, sintetic și sugestiv pentru activitatea de învățare propusă;
- competențele specifice vizate: conform programei școlare în vigoare;
- condițiile necesare desfășurării activității: sunt oferite sintetic detalii despre spațiul în care se desfășoară activitatea propusă, de exemplu laboratorul de informatică sau sala de clasă sau online, alte aspecte necesare pentru organizarea și desfășurarea activității propuse, de exemplu acces la rețeaua Internet, platforme educaționale;
- contextul de învățare: sunt oferite sintetic detalii despre modul de organizare a activității de învățare din perspectiva contextului concret de formare sau dezvoltare a competenței vizate;
- timpul alocat: este indicat, estimativ, timpul necesar pentru realizarea activității;
- forma de organizare a clasei: de exemplu frontal, cu întreaga clasă de elevi, pe grupe;
- metodele didactice utilizate;
- mijloacele de învățământ utilizate (inclusiv link-uri, acolo unde sunt disponibile);
- scenariul activității de învățare, descrierea specifică a acesteia, evidențiindu-se activitatea profesorului și activitatea elevilor: sunt oferite detalii relevante despre modul concret de organizare și desfășurare a activității de învățare:
    - sarcinile de lucru pentru elevi, corelate cu competența specifică vizată;
    - modul de organizare a activității;
    - modalități de evaluare a modului de rezolvare a sarcinilor de lucru;
    - alte elemente care sprijină înțelegerea activității de învățare propuse;

  Scenariul activității de învățare are o sarcină predictivă în ceea ce privește rolul elevului, care se recomandă să fie unul activ-participativ.

- extinderi sau dezvoltări ale activității: sunt oferite detalii despre posibile modalități prin care activitatea de învățare poate fi dezvoltată, de exemplu prin extinderea contextului de formare a competenței vizate, prin sarcini de lucru noi realizate prin teme individuale sau de grup pentru acasă;
- resurse utilizate pentru proiectarea activității: sunt prezentate resurse bibliografice valorificate în dezvoltarea activității de învățare.

Exemplele care urmează sunt adaptabile atât orelor care au loc în sala de clasă, cât și celor desfășurate în mediul online.

3. C. 1. Exemplu de proiectare a unei activități de învățare

Titlu:

Compararea unor algoritmi cunoscuți pentru determinarea celui mai mare divizor comun a două numere naturale din punctul de vedere al numărului de operații elementare realizate

Competențe specifice vizate:

- 4.2. Alegerea unui algoritm eficient de rezolvare a unei probleme

Condiții necesare desfășurării activității:

Activitatea se desfășoară prin interacțiune directă în clasă sau online folosind Teams/Zoom/Meet sau orice instrument de videoconferință, care permite partajarea ecranului.

Include o sarcină de lucru ce va fi realizată în limbajul C++. Elevii vor realiza sarcina de lucru individual și vor putea partaja ecranul cu ceilalți participanți la lecție.

Pentru realizarea sarcinii elevii au nevoie de laptop/desktop sau tabletă.

Contextul de învățare

Este important ca fiecare elev să fie încurajat să participe activ la lecție, să-și exprime ideile și să ofere feedback colegilor și profesorului. O sarcină primită de la profesor care este finalizată, chiar dacă nu are un grad mare de dificultate, va reprezenta o realizare și orice greșeală apărută în realizarea sarcinii primite îl poate ajuta pe elev să ofere feedback privind înțelegerea activității de predare.

Utilizarea unui mediu prietenos și realizarea unor aplicații interesante va face ca elevii să învețe să programeze în cel mai natural mod.

Timpul alocat:

45 de minute

Forme de organizare a clasei:

Activitate frontală/pe grupe de elevi/individuală

Metode didactice utilizate:

Expunerea, demonstrația, conversația euristică

Mijloace de învățământ utilizate:

Fișă de exerciții, mediu de implementare a algoritmilor în C++

Scenariul activității de învățare

<table>
<tr><th>Activitatea profesorului</th><th colspan="3">Activitatea elevului</th></tr>
<tr><td>1. Profesorul aduce în discuție doi algoritmi pentru calcularea celui mai mare divizor comun: algoritmul cu diferențe (al lui Nicomachus)
$cmmdc(a,b)=\begin{cases} cmmdc(a-b,b), pentru\ a>b \\ cmmdc(a,b-a), pentru\ b>a \\ a, pentru\ a=b \end{cases}$
respectiv algoritmul lui Euclid
$cmmdc(a,b)=\begin{cases} a, pentru\ b=0 \\ cmmdc(b,a\%b), pentru\ b\neq 0 \end{cases}$</td><td colspan="3">Elevii participă activ la actualizarea informațiilor privind cei doi algoritmi.</td></tr>
<tr><td rowspan="4">2. Profesorul transmite elevilor sarcina de a calcula numărul de iterații pe care fiecare din cei doi algoritmi le execută pentru calcularea cmmdc pentru datele de intrare: 162 și 16.
Se recomandă parcurgerea pașilor celor doi algoritmi (până când a și b au, din nou, aceleași valori) și determinarea numărului de repetiții.</td><td colspan="2">În prima etapă un elev va exemplifica pentru ceilalți colegi iterațiile pe care le va executa algoritmul cu diferențe
cât timp a>b</td><td>Un elev va exemplifica pentru ceilalți colegi prima iterație pe care o va executa algoritmul lui Euclid</td></tr>
<tr><td>a</td><td>b</td><td rowspan="3">r←162%16
a←16
b←2</td></tr>
<tr><td>162</td><td>16</td></tr>
<tr><td>(1)146
(2)130
(3)114
(4)98
(5)82
(6)66
(7)50
(8)34
(9)18
(10)2</td><td></td></tr>
<tr><td colspan="2">În total 10 iterații</td><td>Prima iterație</td></tr>
</table>

<table>
<tr><td>3. Profesorul întreabă elevii: Ce reprezintă aceste iterații?</td><td colspan="3">Răspunsul așteptat de la elevi este:
La algoritmul cu diferențe se face câte o iterație de fiecare dată când b este cuprins în a, deci numărul de iterații reprezintă chiar câtul împărțirii lui a la b.
Se face practic împărțire prin scăderi repetate.
Primei iterații din algoritmul lui Euclid îi corespund, pentru acest exemplu, 10 iterații în algoritmul cu diferențe.</td></tr>
<tr><td rowspan="4">4. Se recomandă continuarea parcurgerii celor doi algoritmi până când a și b au, din nou, aceleași valori, sau b ajunge 0.</td><td colspan="2">Pentru algoritmul cu diferențe, în această a doua etapă, un elev va exemplifica iterațiile care se vor executa
cât timp b>a</td><td>Pentru algoritmul lui Euclid, un elev va exemplifica a doua iterație care se va executa</td></tr>
<tr><td>a
2</td><td>b
16
(1)14
(2)12
(3)10
(4)8
(5)6
(6)4
(7)2</td><td>r←16%2
a←2
b←0</td></tr>
<tr><td colspan="2">În total 7 iterații</td><td>A doua iterație</td></tr>
<tr><td>5. Profesorul întreabă elevii:
Cum comentați această a doua etapă?</td><td colspan="3">Răspunsul așteptat de la elevi este:
În algoritmul cu diferențe se fac repetiții de câte ori b este mai mare decât a, iar pentru a obține a=b, numărul de iterații este câtul împărțirii lui b la a mai puțin 1. Pentru a doua (și ultima) iterație din algoritmul lui Euclid se fac, pentru acest exemplu, 7 iterații în algoritmul cu diferențe.</td></tr>
</table>

6. Concluziile activității efectuate de elevi: pentru fiecare iterație de determinare a restului împărțirii în ***algoritmul lui Euclid*** se fac, în ***algoritmul cu diferențe***, un număr de iterații egal cu cu câtul fiecărei împărțiri. Elevii trebuie să precizeze care dintre cei doi algoritmi are un număr de iterații mai mic pentru aceleași date de intrare, în consecință care dintre cei doi algoritmi este mai eficient din punctul de vedere al timpului de execuție.

Extinderi sau dezvoltări ale activității (opțional)

Exersare: Implementați cei doi algoritmi în limbaj de programare, folosind câte un contor pentru numărarea iterațiilor. Afișați la final numărul de iterații și comparați-l cu cel așteptat. Precizați numărul de iterații care se fac, pentru fiecare algoritm, pentru următoarele perechi de valori: **1)** `a=2000000000, b=1` **2)** `a=48, b=56` 3) `a=162, b=26` Identificați, în mediul de programare utilizat, timpul de executare al fiecărui algoritm pentru primul set de date de intrare.	*Răspuns*: ***1)*** `1999999999` iterații pentru ***algoritmul lui Nicomachus*** o iterație pentru ***algoritmul lui Euclid*** ***2)*** `6` iterații pentru ***algoritmul lui Nicomachus*** `3` iterații pentru ***algoritmul lui Euclid*** ***3)*** `12` iterații pentru ***algoritmul lui Nicomachus*** `3` iterații pentru ***algoritmul lui Euclid***

Resurse utilizate:

Se pot utiliza următoarele resurse:

1. ***cmmdc - Algoritmul lui Euclid*** aflat la adresa https://www.pbinfo.ro/articole/73/algoritmul-lui-euclid
2. https://ro.wikipedia.org/wiki/Algoritmul_lui_Euclid

3. C. 2. Exemplu de proiectare a unei activități de învățare

Titlu:

Descompunerea unui număr în factori primi

Competențe specifice vizate:

- 3.2. Reprezentarea algoritmilor în pseudocod

Condiții necesare desfășurării activității:

Activitatea de învățare se desfășoară prin interacțiune directă în clasă, sau online, utilizând o platformă care permite întâlniri sincron.

Elevii și cadrul didactic vor avea acces la un calculator/laptop/o tabletă având conexiune stabilă la Internet de bandă largă.

Timpul alocat:

45 de minute

Forme de organizare a clasei:

Frontal, grupe de elevi, individual

Metode didactice utilizate:

Conversația euristică, problematizarea, algoritmizarea

Mijloace de învățământ utilizate:

Fișă de exerciții

Scenariul activității de învățare

Activitatea profesorului	***Activitatea elevului***
Profesorul va începe întâlnirea cu elevii clasei a IX-a, apoi va partaja o aplicație de tip tablă interactivă online. Profesorul transmite elevilor sarcina de a descompune în factori primi numărul 360, utilizând metoda studiată la disciplina matematică.	Un elev va descompune în factori primi numărul **n=360** utilizând metoda matematică El va scrie că **n=360=2^3*3^2*5^1**.
Profesorul va implementa în pseudocod pașii efectuați anterior de elev prin metoda matematică.	
Profesorul va adresa întrebarea: Care sunt variabilele utilizate în momentul descompunerii în factori primi?	Elevii vor preciza variabilele care sunt necesare pentru implementarea algoritmului:

<table>
<tr><td>Profesorul va specifica variabilele necesare:
citește n (număr natural,n>1)
d←2;p←0</td><td>• numărul n,
• d-divizorul prim,
• p exponentul puterii divizorului prim.</td></tr>
<tr><td>Profesorul va adresa întrebarea:
Care este condiția de oprire a algoritmului?
Profesorul va adresa întrebarea: De ce n a devenit 1?
Profesorul va adresa întrebarea:
Ce structură folosim pentru a realiza aceleași instrucțiuni de mai multe ori?</td><td>Elevii vor observa că algoritmul se oprește atunci când n devine 1
Un elev va răspunde că a fost împărțit în mod repetat la divizorii săi primi.
Un elev va răspunde că putem folosi o structură repetitivă (cu test inițial sau cu test final).</td></tr>
<tr><td colspan="2">Profesorul va specifica următorul pas:
┌cât timp n>1 execută
│ ......
└■</td></tr>
<tr><td>Profesorul va adresa întrebarea:
În ce condiții am efectuat împărțirea lui n la d?
Profesorul va adresa întrebarea:
Ce se întâmplă cu puterea divizorului?</td><td>Răspunsul așteptat este: Cât timp restul împărțirii lui n la d este 0, n se va împărți la d.
Răspunsul așteptat este: Exponentul divizorului p crește în momentul în care se realizează o împărțire.</td></tr>
<tr><td colspan="2">Algoritmului i se adaugă următoarele instrucțiuni:
p←0
┌cât timp n>1 execută
│ ┌cât timp n%d=0 execută
│ │ n←[n/d]
│ │ p←p+1
│ └■
└■</td></tr>
<tr><td>Profesorul va adresa întrebarea:
Când îl afișăm pe d?</td><td>Răspunsul așteptat de la elevi este: doar dacă exponentul puterii sale va fi mai mare decât 0.</td></tr>
<tr><td colspan="2">Profesorul va completa algoritmul:
p←0
┌cât timp n>1 execută
│ ┌cât timp n%d=0 execută
│ │ n←[n/d]
│ │ p←p+1
│ └■
│ ┌dacă p>0 atunci
│ │ scrie d
│ └■
└■</td></tr>
</table>

Profesorul va adresa întrebarea: În momentul în care **n** nu se va mai împărți la **d**, ce se întâmplă?	Răspunsul așteptat de la elevi este: algoritmul va căuta următorul divizor, iar exponentul puterii se reinițializează cu **0**.

Profesorul va completa algoritmul:

```
p←0
┌cât timp n>1 execută
│    ┌cât timp n%d=0 execută
│    │     n←[n/d]
│    │     p←p+1
│    └■
│   ┌dacă p>0 atunci
│   │ scrie d
│   └■
│   d←d+1
│   p←0
└■
```

Extinderi sau dezvoltări ale activității (opțional)

Fixarea cunoștințelor: Profesorul solicită elevilor să scrie algoritmii corespunzători următoarelor cerințe:

1.Să se afișeze numărul de factori primi care apar în descompunerea în factori primi a unui număr.

Analizând enunțul, elevii vor enunța necesitatea adăugării unei variabile noi, care va contoriza numărul de factori primi.

Rezolvarea poate fi:

```
citește n (număr natural,n>1)
d←2;p←0;k←0
 ┌cât timp n>1 execută
 │    ┌cât timp n%d=0 execută
 │    │     n←[n/d]
 │    │     p←p+1
 │    └■
 │   ┌dacă p>0 atunci
 │   │ k←k+1
 │   └■
 │   d←d+1
 │   p←0
 └■
scrie k
```

2.Să se afișeze puterea cea mai mare a unui factor prim, din descompunerea în factori primi a unui număr natural **n**.

Analizând enunțul, elevii vor specifica necesitatea adăugării a unei variabile noi, pentru a reține puterea maximă.

Rezolvarea poate fi:

```
citește n (număr natural,n>1)
d←2;p←0;maxp←p
 ┌cât timp n>1 execută
 │    ┌cât timp n%d=0 execută
 │    │     n←[n/d]
 │    │     p←p+1
 │    └■
 │   ┌dacă p>pmax atunci
 │   │ pmax←p
 │   └■
 │   d←d+1
 │   p←0
 └■
scrie pmax
```

3.Să se afișeze puterea cea mai mare a unui factor prim și factorul prim cel mai mare care apare la această putere, în descompunerea în factori primi a unui număr natural **n** citit.

Analizând enunțul, elevii vor specifica necesitatea adăugării a unei variabile noi (față de algoritmul prezentat în problema 2), una care reține factorul prim cel mai mare care are puterea maximă.

Rezolvarea poate fi:

```
citește n (număr natural,n>1)
d←2;p←0;maxp←p;fmax←d
┌cât timp n>1 execută
│    ┌cât timp n%d=0 execută
│    │     n←[n/d]
│    │     p←p+1
│    └■
│   ┌dacă p>=pmax atunci
│   │ pmax←p;fmax←d
│   └■
│   d←d+1
│   p←0
└■
scrie fmax,pmax
```

Profesorul va propune ca sarcină de lucru individuală, scrierea algoritmilor corespunzători rezolvării următoarelor probleme:

1. Determinați puterea cea mai mică a unui factor prim, care apare în descompunerea numărului natural **v**;
2. Afișați cel mai mic factor prim, care are puterea cea mai mică și apare în descompunerea numărului natural citit **m**.

Resurse utilizate:

Pot fi accesate următoarele adrese web pentru aprofundarea cunoștințelor:

1. https://www.pbinfo.ro/articole/75/descompunerea-in-factori-primi
2. https://ro.wikipedia.org/wiki/Num%C4%83r_prim

Alte aspecte utile de împărtășit cu privire la realizarea activității de învățare

Se recomandă implicarea activă a elevilor în găsirea algoritmului principal.

Profesorul va conduce activitatea de învățare utilizând diferite metode (conversația, problematizarea) și ajutând elevii să deslușească singuri pașii de rezolvare a problemei.

În cazul în care elevii nu au înțeles bine algoritmul, în momentul de consolidare a cunoștințelor, vor fi propuse probleme mai ușoare ca de exemplu:

1. Afișați toți divizorii primi ai unui număr natural.

2. Afișați toți factorii primi ai unui număr natural **n**, precum și puterile la care apar acești factori în descompunerea lui **n** în factori primi.

3. C. 3. Exemplu de proiectare a unei activități de învățare

Titlu:

Structuri de control folosite în realizarea algoritmilor - metodă didactică modernă, într-o lecție desfășurată față în față/online

Competențe specifice vizate:

- 1.2. Recunoaşterea situaţiilor în care este necesară prelucrarea algoritmică a informaţiilor
- 3.3. Respectarea principiilor programării structurate în procesul de elaborare a algoritmilor

Condiții necesare desfășurării activității:

A. Activitatea se desfășoară în sala de clasă (harta conceptuală se realizează la tablă/pe hârtie) sau în laboratorul de informatică dacă se utilizează o aplicație specializată pentru construcția hărților conceptuale (exemple de aplicații pentru hărți conceptuale sunt bubbl.us, Lucidchart, Coggle, MindMup).
B. Etapele care implică dialogul sincron cu toți elevii clasei se desfășoară pe o platformă online cum ar fi: Meet, Microsoft Teams, Zoom, Webex sau oricare altă platformă care permite dialogul sincron online. Pentru realizarea hărții se mai poate utiliza o aplicație colaborativă specializată (de exemplu Jamboard).
C. Elevii primesc în formă scrisă indicații de realizare, eventual profesorul trasează începutul hărții (etapa 1 și etapa 2), elevul completând elementele rămase, conform indicațiilor.

Timpul alocat:

45 de minute

Forme de organizare a clasei:

Frontal, grupe de elevi, individual

Metode didactice utilizate:

Metoda Hărților conceptuale, Conversația dirijată

Mijloace de învățământ utilizate:

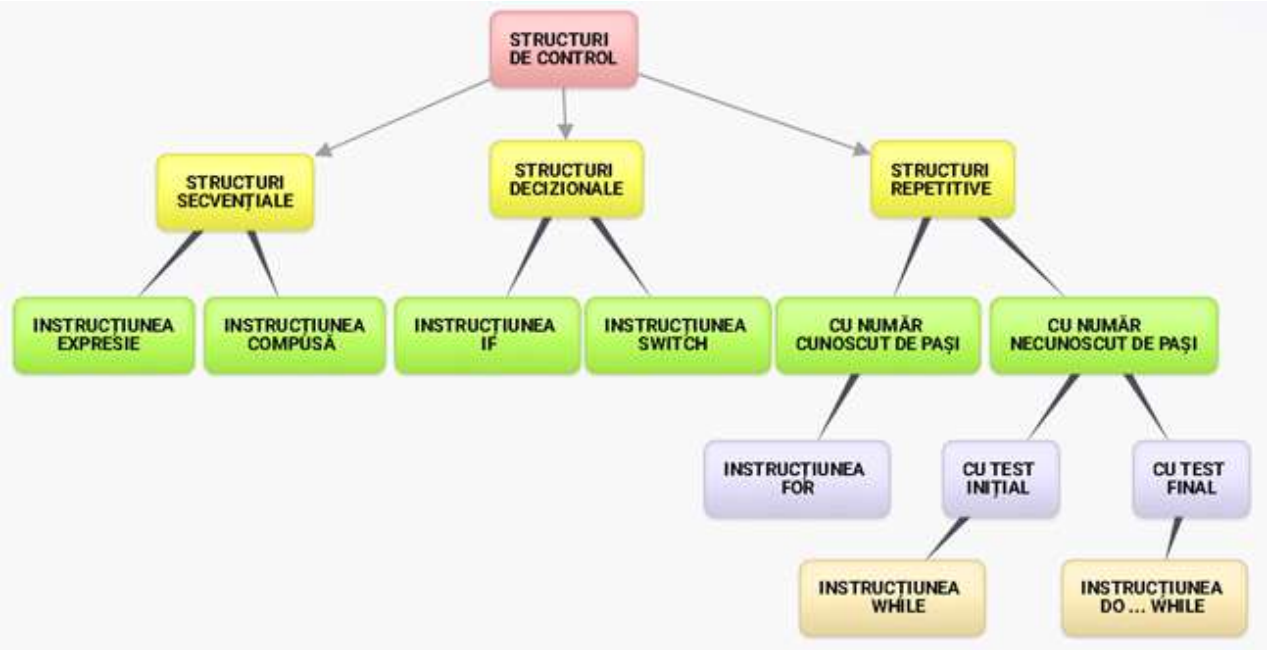
Fișă de exerciții, aplicații pentru hărți conceptuale, aplicații colaborative, aplicația Code Blocks sau o aplicație care să conțină un compilator online pentru limbajul C++, aplicația Microsoft Teams/Zoom/Meet.

Scenariul activității de învățare

Activitatea profesorului	***Activitatea elevului***
Profesorul propune realizarea unei hărți conceptuale, o modalitate prin care informațiile vor fi organizate logic, prin evidențierea relațiilor existente între diferite concepte. Profesorul prezintă informații despre harta conceptuală, care permite: -o reprezentare grafică a componentelor unor concepte vizualizând și relațiile existente între ele; -se pornește de la un termen cheie, iar organizarea poate fi arborescentă sau de jur împrejurul conceptului cheie; -noțiunile care apar în hartă sunt structurate și în acest fel pot fi mai bine înțelese. Modul de realizarea a unei hărți conceptuale poate fi unul dirijat de profesor sau lăsat la alegerea elevilor.	

Propunerea profesorului este de a realiza o hartă conceptuală având drept cuvânt cheie: STRUCTURI DE CONTROL. Profesorul indică elevilor modalități de realizare a hărții: În mediul online există aplicații cu ajutorul cărora se pot realiza hărți conceptuale. Dacă nu există posibilitatea utilizării unei aplicații specializate pentru realizarea unei hărți conceptuale, atunci harta poate fi realizată pe tablă sau pe un suport flipchart sau pur și simplu pe o hârtie. Metoda folosită de profesor este *conversația dirijată*.	**Etapa 1**: Pornind de la STRUCTURI DE CONTROL, elevii, organizați pe grupe, își notează idei și gânduri, nu neapărat ierarhizate pornind de la cuvântul cheie propus. **Etapa 2**: Elevii vor grupa elementele pe care le-au adunat în funcție de importanța lor, de cât de relevante sunt, de modul în care se pot folosi. **Etapa 3**: Elementele ce definesc STRUCTURI DE CONTROL pot fi: -structuri liniare; -structuri alternative; -structuri repetitive. Pornind de la elementul cheie propus, elevii vor trasa linii care să indice relaționarea acestor elemente. **Etapa 4**: Stabilirea pentru fiecare concept propus la etapa 3 a unor exemple de structuri care pot continua modelarea hărții conceptuale. **Etapa 5**: Dacă se lucrează pe grupe în Jamboard, atunci toți elevii pot face Turul Galeriei vizualizând ce a realizat fiecare grupă. **Etapa 6: Exercițiul poate continua.** Elevii primesc o fișă de lucru cu exemple de probleme. Ei trebuie să rezolve fișa.

Un model de hartă conceptuală realizat cu bubble.us poate fi:



Fișa de lucru - cu exemple de probleme:

Profesorul propune o fișă de lucru cu **10** probleme numerotate cu coduri de la **P1** la **P10**. Se cere încadrarea lor pe tipuri de structuri de control folosite în algoritmul care le rezolvă. Dacă un algoritm corespunzător unei probleme poate utiliza mai multe structuri de control, atunci se specifică acest lucru prin trasarea liniilor care să indice relaționarea codului problemei cu structurile de control folosite. Pornind de la STRUCTURI DE CONTROL, elevii, organizați pe grupe, trasează linii care indică relaționarea codului problemei cu structura de control folosită.

P1 – Se citește de la tastatură un număr natural **n**. Să se calculeze și să se afișeze oglinditul numărului **n**. Oglinditul se obține considerând cifrele numărului în ordine inversă (de la dreapta la stânga).

P2 – Se citește de la tastatură un număr natural **n**. Să se calculeze și să se afișeze numărul de cifre ale lui **n**.

P3 – Se citește de la tastatură un număr natural **n(n<10)**. Să se calculeze și să se afișeze valoarea produsului factorial, notat cu **n!** și obținut astfel: **n!=1*2*...*n**.

P4 – Se citește de la tastatură un număr întreg **x**. Să se calculeze și să se afișeze modulul numărului **x**.

P5 – Se citește de la tastatură un număr întreg **x**. Să se testeze dacă **x** este un număr par și să se afișeze un mesaj corespunzător.

P6 – Se citesc de la tastatură coeficienții reali ai unei ecuații de gradul al II-lea, de forma $a*x^2+b*x+c=0 \quad (a\neq 0)$. Să se rezolve ecuația în mulțimea numerelor reale și să se afișeze soluțiile, iar dacă ecuația nu are soluții reale, să se afișeze mesajul ***nu are solutii reale***.

P7 – Se citesc de la tastatură **3** numere reale **x, y, z**. Să se verifice dacă aceste numere pot fi laturile unui triunghi. Dacă da, să se afișeze aria triunghiului, iar dacă nu, să se afișeze mesajul ***Nu***.

P8 – Se citesc de la tastatură două numere reale **a** și **b**. Să se determine și să se afișeze cea mai mare valoare dintre cele două numere.

P9 – Se citesc de la tastatură un număr natural nenul **n** și **n** valori reale. Să se calculeze și să se afișeze media aritmetică a valorilor strict pozitive, iar dacă nu sunt astfel de numere să se afișeze valoarea **0**.

P10 – Se citește de la tastatură un număr natural **n**. Să se calculeze și să se afișeze suma cifrelor numărului **n**.

Observații:

Principalele avantaje ale utilizării hărţilor conceptuale:

-facilitează evaluarea structurilor cognitive ale elevilor, cu accent pe relaţiile stabilite între concepte;

-determină elevii să practice o învăţare activă, logică;

-permit profesorului să emită aprecieri referitoare la eficienţa stilului de învăţare al elevilor şi să îi ajute să-şi regleze anumite componente ale acestuia;

-asigură „vizualizarea" relaţiei dintre componenta teoretică şi cea practică a pregătirii elevilor;

-facilitează surprinderea modului în care gândesc elevii, a modului în care îşi construiesc demersul cognitiv, permiţând ulterior diferenţierea şi individualizarea instruirii;

-pot fi integrate cu succes în orice strategie de evaluare.

Valorificarea competențelor formate pe parcursul ciclului gimnazial

Sunt valorificate competențele formate cu privire la introducerea structurilor de control pornind de la necesitatea utilizării acestora în situaţii concrete.

3. C. 4. Exemplu de proiectare a unei activități de învățare

Titlu:

Elaborarea algoritmului pentru determinarea termenilor șirului lui Fibonacci.

Competențe specifice vizate:

- 1.2. Recunoaşterea situaţiilor în care este necesară prelucrarea algoritmică a informaţiilor
- 4.5. Utilizarea unui mediu de programare pentru limbajul C++
- 5.1. Elaborarea unui algoritm de rezolvare a unor probleme din aria curriculară a specializării

Condiții necesare desfășurării activității:

Activitatea se desfășoară prin interacțiune directă, folosind Teams/Zoom/Meet sau orice instrument de videoconferință, care permite partajarea ecranului. Include sarcini de lucru care vor fi realizate în C++. Elevii vor realiza sarcinile de lucru individual, și vor putea partaja ecranul cu ceilalți participanți la lecție. Au nevoie de laptop/desktop sau tabletă pe care să fie instalat Codeblocks sau pot utiliza un compilator online (exemplu: https://www.onlinegdb.com/online_c++_compiler)

Timpul alocat:

40 de minute

Forme de organizare a clasei:

Frontal, grupe de elevi, individual

Metode didactice utilizate:

- conversația dirijată;
- metode didactice moderne/alternative - Tehnica 3-2-1.

Mijloace de învățământ utilizate:

Fișă de exerciții, aplicații colaborative, aplicația Code Blocks sau o aplicație care să conțină un compilator online pentru limbajul C++, aplicația Microsoft Teams/Zoom/Meet.

Scenariul activității de învățare

Activitatea începe cu o discuție euristică (online) referitoare la șirurile de numere.

Profesorul poate adresa întrebări cum ar fi: Ce este un șir de numere? Cum poate fi definit un șir de numere? Când două sau mai multe numere fac parte din același șir?.

Profesorul menționează că lecția este dedicată „explorării" șirului lui Fibonacci. În următoarele 5 minute, partajează ecranul și le descrie elevilor acest șir, fie accesând pagina web https://www.mathsisfun.com/numbers/fibonacci-sequence.html, fie utilizând whiteboard-ul din aplicație.

Pentru a evidenția importanța acestui șir, în următoarele 5 minute, elevii vizualizează împreună cu profesorul filmul care prezintă legătura dintre numerele Fibonacci și natura înconjurătoare. https://www.youtube.com/watch?v=P0tLbl5LrJ8

În următoarele 5 minute, elevii sunt provocați să scrie primele 10 numere Fibonacci. În acest fel profesorul primește un feedback imediat legat de însușirea noțiunii matematice. Pentru verificare , accesează pagina https://www.dcode.fr/fibonacci-numbers și elevii primesc confirmarea corectitudinii calculului efectuat.

Profesorul întreabă elevii care este al 50-lea număr Fibonacci. Este evident că această întrebare va conduce la ideea de a scrie un program care să determine al n-lea număr Fibonacci.

Este și primul program pe care profesorul îl va implementa împreună cu elevii în următoarele 10 minute. Profesorul descrie algoritmul și unul dintre elevi partajează ecranul și scrie programul. Se analizează corectitudinea acestuia.

Următoarele 10 minute sunt alocate pentru modificarea algoritmului scris astfel încât, pentru un n dat, să se verifice dacă este acesta este număr Fibonacci și, în caz afirmativ, să se afișeze poziția pe care se află acesta.

Elevii vor lucra în pereche, având la dispoziție chat-ul aplicației pentru a colabora și a găsi rezolvarea problemei. Echipa care rezolvă prima este invitată să prezinte colegilor algoritmul scris.

În cazul în care elevii întâmpină dificultăți în rezolvarea problemei, profesorul poate oferi indicații sau chiar poate scrie algoritmul lăsând spații libere ce urmează a fi completate cu ajutorul elevilor.

În final, elevii sunt invitați să propună probleme care implică șirul lui Fibonacci (de exemplu: cel mai mare număr Fibonacci mai mic decât o valoare dată, al n-lea număr Fibonacci impar etc.). Aceste probleme vor fi scrise pe chat și salvate de către profesor într-un fișier ce va reprezenta punctul de plecare pentru ora următoare.

Pentru evaluare, profesorul aplică Tehnica 3-2-1. Elevii sunt invitați să identifice:

- 3 termeni (concepte) învăţate;
- 2 idei despre care ar dori să înveţe mai mult în continuare;
- o capacitate, o pricepere sau o abilitate pe care consideră ei că au dodândit-o în urma activităţii de predare-învăţare.

Ca temă pentru lucru individual, li se propune elevilor să rezolve un test (de exemplu https://quizizz.com/admin/quiz/591b092dd94358100065c0a7/fibonacci), ca o provocare de a descoperi informații interesante referitoare la marele matematician și la șirul care-i poartă numele.

Observații:

Este important ca fiecare lecție să încerce să facă legătura cu realitatea. Fiecare elev să fie încurajat să participe activ la lecție să-și exprime ideile și să ofere feedback colegilor și profesorului. Elevii trebuie să înțeleagă că, orice greșeală apărută în proiectarea unei aplicații poate fi corectată.

Utilizarea unui mediu prietenos și realizarea unor aplicații interesante va face ca elevii să învețe să programeze în cel mai natural mod și să înțeleagă rolul programării în viața reală.

Resurse:

https://www.youtube.com/watch?v=P0tLbl5LrJ8
https://www.onlinegdb.com/online_c++_compiler
https://www.mathsisfun.com/numbers/fibonacci-sequence.html
https://quizizz.com/admin/quiz/591b092dd94358100065c0a7/fibonacci

4. ADAPTAREA LA PARTICULARITĂȚILE/CATEGORIILE DE ELEVI ÎN SITUAȚII DE RISC

Dacă profesorul identifică în cadrul colectivului de elevi persoane provenite din medii sau grupuri dezavantajate, elevi cu cerințe educaționale speciale, elevi cu statut socio-economic scăzut, cu părinți care au nivel educațional scăzut, elevi cu rezultate scăzute sau foarte scăzute la învățătură, atunci va trebui să aibă în vedere aplicarea unor activități de învățare remediale, în urma cărora elevii în cauză să reușească să depășească statutul de elevi în situații de risc și să aibă un parcurs educațional corespunzător.

Sunt prezentate mai jos unele exemple de activități de învățare remediale care se recomandă să fie reluate, eventual în mod repetat, în contexte diferite, în cazul semnalării unor situații de risc.

Competența specifică clasa a IX-a	*Exemple de activități de învățare*
IX MII 4.5 Utilizarea unui mediu de programare	• realizarea codului sursă prin transcrierea structurilor algoritmului cu ajutorul instrucțiunilor/comenzilor specifice limbajului, evidențiind facilitățile mediului de programare pentru alinierea instrucțiunilor, relaționarea unor părți ale instrucțiunilor (if-else, do-while, acolade) semnalarea unor erori de sintaxă pe măsura scrierii codului; • rularea linie cu linie sau prin salt la o anumită linie a unei surse existente pentru urmărirea valorilor variabilelor din program, respectiv pentru identificarea erorilor de implementare sau de proiectare a algoritmului în vederea depanării codului sursă.

IX MI/SN 3.1 Analizarea enunţului unei probleme: identificarea datelor de intrare şi a datelor de ieşire (cu specificarea tipului datelor şi a relaţiilor existente între date) şi stabilirea paşilor de rezolvare a problemei; IX MII 3.1 Analizarea enunţului unei probleme şi stabilirea paşilor de rezolvare a problemei.	• descrierea, pas cu pas, a rezolvării unei probleme și divizarea ei în etape/probleme mai simple care, la rândul lor, sunt analog divizate.
IX MI/SN, IX MII 3.2 Reprezentarea algoritmilor în pseudocod.	• argumentarea utilizării pseudocodului ca o formă de reprezentare a unui algoritm independentă de tehnologia/limbajul de programare utilizat, ca punct de pornire pentru discuții în cadrul unui proiect complex; • executarea pas cu pas a unor secvenţe de algoritm, urmărind evoluția variabilelor și analizând efectele executării structurilor de control; • adaptarea unor algoritmi existenți în scopul rezolvării unor probleme asemănătoare; • înlocuirea diferitelor tipuri de structuri repetitive și de structuri de decizie, prin construirea unui algoritm echivalent cu un algoritm dat.
IX MI/SN 4.1, IX MII 5.1 Elaborarea unui algoritm de rezolvare a unor probleme din aria curriculară a specializării.	• identificarea unor aplicații la disciplina matematică și realizarea algoritmilor de rezolvare precum și implementarea programelor corespunzătoare în limbaj de programare, de exemplu, calculul termenilor unei progresii aritmetice, calculul suprafețelor diferitelor figuri din geometria plană, rezolvarea ecuațiilor liniare, rezolvarea unui sistem de două ecuații cu două necunoscute, poziția relativă a două cercuri în plan, determinarea intersecției mai multor intervale de valori.

IX MI/SN 3.1 Analizarea enunţului unei probleme: identificarea datelor de intrare şi a datelor de ieşire (cu specificarea tipului datelor şi a relaţiilor existente între date) şi stabilirea paşilor de rezolvare a problemei. IX MII 3.1 Analizarea enunţului unei probleme şi stabilirea paşilor de rezolvare a problemei.	• scrierea unui algoritm pentru rezolvarea unei probleme date care utilizează, în mod repetat, o secvență de prelucrare, în vederea identificării etapelor independente, precum și a șabloanelor, caracteristicilor repetitive, de exemplu: determinarea celui mai mare divizor comun a două numere naturale, determinarea sumei cifrelor unui număr natural care are cel puțin 3 cifre.
IX MI/SN, IX MII 3.3. Respectarea principiilor programării structurate în procesul de elaborare a algoritmilor.	• scrierea în pseudocod a unor algoritmi elementari și urmărirea pas cu pas a execuției acestora, evidențiind unele cazuri particulare tratate corespunzător și siguranța în funcționarea algoritmului, de exemplu: prelucrare a cifrelor unui număr, divizibilitate, prelucrare a unor secvențe de valori.
IX MII 4.1. Transcrierea algoritmilor din pseudocod într-un limbaj de programare.	• prezentarea unor exemple de implementare în mai multe limbaje de programare a unor algoritmi elaborați și executarea acestora pe calculator, evidențiind prezența unor elemente reper în sintaxa instrucțiunilor de bază ale fiecăruia dintre acestea, precum și diferențe ale scrierii acestor instrucțiuni.

Elevii care nu beneficiază de un computer/laptop personal pentru exersarea implementării programelor și realizarea temelor pot utiliza telefonul inteligent sau tableta (având costuri mai mici) pentru a accesa gratuit, online, aplicații care permit operații de editare, compilare și executare a programelor. Pentru astfel de elevi, profesorul poate adapta temele astfel încât ele să poată fi scrise și pe caiete, urmând ca în cadrul școlii să se urmărească implementarea lor pe calculator, selectiv.

De asemenea, unitatea de învățământ poate stabili un orar de utilizare a laboratorului de informatică de către elevii care nu au acces de acasă la tehnologiile specifice.

O altă provocare a profesorului de informatică în activitatea desfășurată la clasa a IX-a o reprezintă elevii care vin din ciclul gimnazial însoțiți de un bagaj complex de achiziții, elevi care

au participat la concursuri școlare în domeniul informaticii. Pentru aceștia, profesorul va trebui să adapteze și să nuanțeze conținuturile astfel încât elevul să nu ajungă pe o pantă descendentă.

Recomandăm astfel organizarea unor activităţi de învăţare complexe, concretizate, de exemplu, prin realizarea unei aplicaţii software în echipă, precum un site web sau o aplicație pentru un dispozitiv mobil.

Un real folos în activitatea de la clasă îl reprezintă paginile web specializate atât pentru pregătirea la nivelul unei clase de nivel mediu, cât și pentru pregătirea concursurilor de informatică, unde pot fi găsite atât resurse educaționale, cât și cerințe pentru a căror rezolvare se oferă feedback imediat. Itemii acoperă competențele și conținuturile din programele de informatică ale ciclului liceal și sunt cu alegere multiplă, cu răspuns scurt și de tipul rezolvare de probleme.

Itemii propuși, de nivel cognitiv scăzut sau mediu, pot fi utilizați pentru început de către elevii aflați în categorii de risc, care dovedesc lacune în asimilarea competențelor de algoritmică și programare din ciclul gimnazial. De asemenea, ei pot utiliza ca suport resursele educaționale care se regăsesc pe astfel de site-uri.

Itemii de nivel cognitiv ridicat (probleme dificile, sau utilizate în concursuri) pot fi un exercițiu bun pentru elevii care stăpânesc cunoștințele de algoritmică și programare din gimnaziu și care se îndreaptă spre concursurile școlare.

Mai multe exemple de astfel de site-uri specializate au fost furnizate în capitolele anterioare.

BIBLIOGRAFIE

1. Bocoș, M., Jucan, D., Teoria și metodologia instruirii. Teoria și metodologia evaluării: repere și instrumente didactice pentru formarea profesorilor, Ed. Paralela 45, 2019
2. Brut, M., Instrumente pentru e-learning, Ed. Polirom, 2006
3. Cerchez, E., Șerban, M., Programarea în limbajul C/C++ pentru liceu, vol I-IV, Ed. Polirom, 2004-2013
4. Cerghit, I., Metode de învățământ, Ed. Polirom, 2006
5. Cerghit, I., Sisteme de instruire alternative și complementare. Structuri, stiluri și strategii, Ed. Polirom, 2008
6. Cormen, T., Leiserson, Ch., Rivest, R. Introducere în algoritmi, Ed. Byblos, Cluj, 2004
7. Cucoș, C., Pedagogie, Ed. Polirom, 2014
8. Cucoș, C., Psihopedagogie pentru examenele de definitivare și grade didactice, Ed. Polirom, 2016
9. Cucoș, C., Teoria si metodologia evaluării, Ed. Polirom, 2008
10. Gălățan, C., Secrete C++, Ed. Microinformatica, 2006
11. Giumale, C., Negreanu, L., Călinoiu, S., Proiectarea și analiza algoritmilor. Algoritmi de sortare, Ed. ALL, 1997
12. Ionescu, M., Bocoș, M.D. (coord.), Tratat de didactică modernă, Ed. Paralela 45, 2017
13. Jinga, I., Istrate, E., Instruirea și evaluarea asistată de calculator, Ed. ALL, 2006
14. Knuth, D.E., Arta programării calculatoarelor vol.2, Algoritmi seminumerici, Ed. Teora, 2000
15. Knuth, D.E., Arta programării calculatoarelor, vol.1, Algoritmi fundamentali, Ed. Teora, 1999
16. Knuth, D.E., Arta programării calculatoarelor, vol.3, Sortare și căutare, Ed. Teora, 2001
17. Manolescu, M., Evaluarea școlară, Ed. Meteor, 2006
18. Masalagiu, C., Asiminoaie, A., Țibu, M., Didactica predării informaticii, Ed. Polirom, 2016
19. Mitrana, V., Provocarea algoritmilor, Ed. Agni, 1994
20. Oprea, C.L., Strategii didactice interactive, Ed. Didactică și pedagogică, 2006

21. Petre, C., Popa, D. ș.a., Metodica predării Informaticii și Tehnologiei Informației, Ed. Arves, 2002
22. Potolea, D., Neacșu, I., Iucu, R.B., Pânișoară, I.O. (coord), Pregătirea psihopedagogică. Manual pentru definitivat și gradul didactic II, Ed. Polirom, 2008
23. Stoica, A. (coord.), Evaluarea curentă și examenele, Ghid pentru profesori, Ed. Prognosis, 2001
24. Stoica, A., Evaluarea progresului școlar. De la teorie la practică. Ed. Humanitas, 2003

*** Didactica formării competenţelor. Cercetare –dezvoltare – inovare – formare, Universitatea de Vest „Vasile Goldiş” din Arad, Centrul de Didactică şi Educaţie, ”Vasile Goldiş” University Press Arad, 2012

*** Ghid de evaluare la informatică și tehnologia informației, Serviciul Național de Evaluare și Examinare, Ed. Aramis Print, 2001

*** Programul național de Dezvoltare a Competențelor de Evaluare ale Cadrelor Didactice (DeCeE), Centrul Național pentru Curriculum și Evaluare în Învățământul Preuniversitar - CNCEIP, Ed. Euro Standard, 2008

Coordonator:

Livia Demetra ȚOCA

Monitorizare:

Nușa DUMITRIU-LUPAN

Adrian IFTENE

Colectiv de autori:

Cristina Elena ANTON

Alina Gabriela BOCA

Adriana CĂLINOIU

Diana CONTRAȘ

Cătălina-Anca ENESCU

Maria-Gabriela IONESCU

Eugenia-Cristiana IORDAICHE

Cornelia MAIER

Aidan NURLA

Roxana-Gabriela TÎMPLARU

Florentina UNGUREANU