Cadrul de referință pentru Olimpiada Raională/Municipală/Zonală și Republicană la Informatică

Cuprins

Preambul

Olimpiada Republicană la Informatică se organizează de Ministerul Educației și Cercetării și Agenția Națională pentru Curriculum și Evaluare, în colaborare cu Consiliul Olimpic Republican, cu Universitatea Tehnică a Moldovei, cu Universitatea de Stat din Moldova și organele locale de specialitate în domeniul învățământului.

În conformitate cu Regulamentul privind concursurile și olimpiadele școlare, aprobat prin Ordinul nr.1222/2020, Olimpiada Republicană la Informatică are drept scop identificarea, stimularea și promovarea elevilor capabili de performanțe înalte în domeniul informaticii, prin valorificarea potențialului intelectual, a gândirii algoritmice și a competențelor de rezolvare a problemelor.

Olimpiada contribuie la formarea și dezvoltarea competențelor de analiză, abstractizare, modelare și implementare algoritmică, având totodată rolul de a asigura continuitatea pregătirii elevilor pentru competițiile naționale și internaționale de profil.

Principii de organizare și evaluare a probei de concurs

Proba de concurs la disciplina Informatică se organizează în conformitate cu principiile generale stabilite prin actele normative în vigoare privind desfășurarea olimpiadelor școlare, având în vedere specificul disciplinei și particularitățile evaluării algoritmice.

Testul unic

La etapa raională/municipală/zonală Olimpiada la Informatică se desfășoară în baza testelor unice, în conformitate cu ordinele Ministerului Educației și Cercetării.

În contextul disciplinei Informatică, testul de concurs nu se limitează la enunțul problemelor, ci reprezintă un ansamblu unitar, care include:

- enunțurile problemelor;
- seturile de teste utilizate pentru evaluare;
- mecanismele de verificare automată.

Aceste componente sunt inseparabile și trebuie tratate ca un tot unitar pentru a asigura corectitudinea și echitatea evaluării.

Evaluarea automată și centralizată

Având în vedere faptul că evaluarea soluțiilor la Informatică se realizează prin rularea automată a programelor pe seturi de teste standardizate, evaluarea automată constituie parte integrantă a testului de concurs, și nu un instrument auxiliar.

În acest context, elaborarea enunțurilor, definirea seturilor de teste și asigurarea mecanismelor de evaluare automată se realizează în mod centralizat de Consiliul Olimpic Republican, iar gestionarea platformei informatice de concurs este asigurată integral de către Agenția Națională pentru Curriculum și Evaluare și Consiliul Olimpic Republican.

Această abordare este esențială în special pentru etapa raională/municipală/zonală, organizată în baza testelor unice, în vederea asigurării unității de evaluare, comparabilității rezultatelor și respectării principiului echității competiționale.

Responsabilități instituționale

Centralizarea conceperii și evaluării probei de concurs nu afectează responsabilitățile instituționale privind organizarea și desfășurarea probei la nivel local.

Organele locale de specialitate în domeniul învățământului și instituțiile de învățământ desemnate ca centre de concurs rămân responsabile de asigurarea condițiilor organizatorice, tehnice și de supraveghere a competitorilor, în conformitate cu actele normative în vigoare.

Integritate și securitate

Pe durata probei de concurs, competitorii pot utiliza exclusiv instrumentele permise pentru lucrul la calculator, fiind interzis accesul la materiale auxiliare, dispozitive de comunicare sau resurse externe care ar putea afecta caracterul obiectiv al evaluării.

Organizarea probei la nivel local presupune asigurarea condițiilor logistice și tehnice adecvate, inclusiv utilizarea unor echipamente informatice configurate corespunzător, astfel încât fiecare competitor să lucreze în mod individual.

De asemenea, desfășurarea probei este asigurată prin supraveghere adecvată, realizată de personal desemnat, în condiții care garantează imparțialitatea și evitarea conflictelor de interese, în conformitate cu reglementările aplicabile.

Instrucțiuni specifice de organizare

Descrierea probei de concurs

1. Proba de concurs la disciplina Informatică constă în rezolvarea a trei probleme de natură algoritmică. Pentru fiecare problemă, competitorul elaborează un program de calculator care să rezolve cerința formulată în enunț.
2. Pentru fiecare problemă este pus la dispoziția competitorilor un enunț, însoțit de o fișă intitulată „Descrierea generală a problemei”, care conține următoarele informații:
    a. denumirea problemei;
    b. tipul problemei (dacă este cazul);

    Notă: Problema poate fi clasică (batch), interactivă sau evaluată prin mecanisme specifice, fiind precizat explicit în enunțul problemei. Ultimele două tipuri se aplică doar la etapa națională sau de selecție pentru competiții internaționale.

    c. restricția referitoare la volumul de memorie utilizat;
    d. restricția referitoare la timpul de execuție;
    e. numărul de teste utilizate pentru evaluare;
    f. numărul de puncte alocate problemei.
3. Subiectele probei de concurs sunt transmise prin intermediul platformei informatice de concurs sau electronic de către Agenția Națională pentru Curriculum și Evaluare, și sunt administrate de către membrii Consiliului Olimpic Republican.
4. Structura și durata probei de concurs diferă în funcție de etapa olimpiadei, după cum urmează:
    a. la etapa raională / municipală / zonală, proba de concurs se desfășoară într-o singură zi și constă în rezolvarea a trei probleme, într-un interval de 4 ore astronomice;
    b. la etapa republicană, proba de concurs se desfășoară pe parcursul a două zile distincte, fiecare zi constând în rezolvarea a trei probleme, într-un interval de 4 ore astronomice.
5. Punctajul final al competitorului la etapa republicană se calculează ca suma punctajelor obținute pentru toate cele șase probleme rezolvate pe parcursul celor două zile de concurs.

Platforma informatică de concurs

6. Platforma informatică utilizată pentru desfășurarea Olimpiadei la Informatică este asigurată, administrată și gestionată integral de către Consiliul Olimpic Republican. Prin intermediul acestei platforme se realizează:
    a. transmiterea subiectelor și a testelor de concurs;
    b. încărcarea și evaluarea automată a soluțiilor transmise de competitori;
    c. generarea clasamentelor;
    d. analiza soluțiilor proprii de către competitori, după încheierea probei de concurs.

    Notă: Analiza soluțiilor proprii se realizează exclusiv după finalizarea probei de concurs și publicarea rezultatelor oficiale și nu influențează evaluarea, punctajele sau clasamentele stabilite.

7. Adresa URL a platformei de concurs, precum și datele de acces aferente, sunt comunicate prin anunțuri separate de organizator, anterior desfășurării probei de concurs.

Interacțiunea cu platforma Olimpiadei de Informatică

8. În ziua de concurs, competitorii rezolvă probleme de natură algoritmică și încarcă programele elaborate pe platforma informatică pusă la dispoziție de organizatori. Pentru fiecare problemă este evaluat un singur fișier-sursă. În cazul în care pentru aceeași problemă sunt încărcate mai multe fișiere-sursă, este luat în considerare fișierul pentru care competitorul a obținut cel mai mare punctaj.
9. Cerințele privind denumirea fișierului-sursă sunt specificate, după caz, în enunțul problemei. Numărul de încărcări și frecvența acestora sunt limitate și sunt afișate în interfața de lucru a competitorului pentru fiecare problemă.
10. Competitorii pot utiliza platforma pentru compilarea și depanarea programelor transmise. În acest scop pot fi folosite atât testele furnizate de organizatori, dacă acestea există, cât și testele proprii ale competitorului. În cazul utilizării testelor proprii, competitorul este responsabil de încărcarea acestora pe platformă, în limitele afișate în interfața de lucru.

    Notă: Testele puse la dispoziție de organizatori au caracter orientativ și nu constituie, în mod necesar, instrumente de evaluare oficială.

11. În programele transmise platformei nu este permisă declararea și utilizarea variabilelor de tip fișier. Citirea datelor de intrare se realizează prin dispozitivul standard de intrare, iar afișarea rezultatelor se face prin dispozitivul standard de ieșire, cu excepția cazului în care în enunțul problemei este specificat explicit altfel.

    Notă. În procesul de dezvoltare și testare pe propriul calculator, competitorul poate utiliza fișiere, cu condiția ca fișierul-sursă transmis serverului să nu conțină astfel de prelucrări.

Mediul de lucru

12. Limbajele de programare acceptate în cadrul Olimpiadei la Informatică sunt Pascal, C (standard C11) și C++ (standard C++17).
13. Competitorilor li se pun la dispoziție medii de lucru care asigură vizualizarea documentelor PDF, accesul controlat la platforma de concurs și dezvoltarea programelor în limbajele admise, utilizând compilatoare compatibile cu standardele FreePascal (ISO 7185), C11 și C++17.
14. Programele transmise pot utiliza facilitățile oferite de compilatorul ales, cu excepția celor care contravin regulilor Olimpiadei. Sunt interzise, inclusiv:
    a. apeluri de întreruperi;
    b. scrierea în porturi;
    c. modificarea sau resetarea timpului sistemului;
    d. lansarea altor programe sau procese;
    e. accesarea rețelei.
15. Competitorilor le este interzisă lansarea sau utilizarea aplicațiilor de configurare și/sau monitorizare a sistemului de operare ori a rețelei. De asemenea, nu este permisă instalarea sau dezinstalarea de programe și nici ștergerea ori modificarea altor fișiere decât cele create de competitor în cadrul probei.

Syllabus

Lista subiectelor stabilește cadrul de referință pentru elaborarea probelor de concurs. Distribuirea tematicilor reflectă o evoluție progresivă a complexității și urmărește alinierea pregătirii elevilor la cerințele competițiilor internaționale de Informatică, contribuind totodată la identificarea și selecția elevilor talentați.

A. Aritmetică și matematică discretă

Secțiunea urmărește dezvoltarea fundamentelor matematice necesare rezolvării problemelor algoritmice, de la aritmetică elementară la noțiuni de matematică discretă cu aplicații în informatică.

Subiecte	**Descriere succintă** *Aritmetică și matematică discretă*	**7-9**	**10**	**11**	**12**[1]
1. Numere întregi: operații, divizibilitate, GCD/LCM	Operații fundamentale cu numere întregi și proprietăți de divizibilitate; determinarea celui mai mare divizor comun și a celui mai mic multiplu comun.	+	+	+	+
2. Numere prime și factorizare	Identificarea numerelor prime și descompunerea numerelor în factori primi, utilizate în probleme de numărare și aritmetică.	+	+	+	+
3. Aritmetică modulară (fără inverse)	Operații de bază în aritmetica modulară (adunare, scădere, înmulțire), cu aplicații în calcul eficient și probleme de resturi.	+	+	+	+
4. Aritmetică modulară cu inverse	Determinarea și utilizarea inverselor modulare în contexte simple, fără tehnici avansate din teoria numerelor.	-	-	+	+
5. Mica Teoremă a lui Fermat	Utilizarea teoremei în probleme de aritmetică modulară și verificări de tip teoretic sau computațional.	-	-	*	*

[1] Notă: Semnificația simbolurilor utilizate în tabel este următoarea:
„ - ” – conținutul nu este prevăzut;
„ + ” – conținutul este inclus la nivel de bază;
„ * ” – conținutul este inclus la nivel avansat (competiție națională sau selecții internaționale).

Subiecte	Descriere succintă *Aritmetică și matematică discretă*	7-9	10	11	12[1]
6. Bazele criptografiei	Noțiuni elementare legate de criptare și securitatea informației, bazate pe proprietăți aritmetice simple.	*	+	+	+

B. Structuri de date

Această secțiune urmărește dezvoltarea capacității de organizare și gestionare eficientă a datelor, prin utilizarea structurilor de date.

Subiecte	Descriere succintă *Structuri de date*	7-9	10	11	12
1. Tablou	Structuri liniare pentru stocarea și accesarea colecțiilor de date numerice, utilizate în probleme de bază și calcule iterative.	+	+	+	+
2. Liste, stivă, coadă	Structuri liniare dinamice și structuri cu acces restricționat, utilizate pentru modelarea proceselor secvențiale și recursive.	+	+	+	+
3. Structuri pentru mulțimi și asocieri	Structuri pentru stocarea elementelor unice sau a perechilor cheie–valoare, cu operații de căutare și actualizare eficiente.	+	+	+	+
4. Heap (coadă de priorități)	Structură ierarhică ce permite acces rapid la elementul cu prioritate maximă sau minimă.	*	+	+	+
5. Union–Find (mulțimi disjuncte)	Structură pentru gestionarea partiționării unei mulțimi în submulțimi disjuncte, cu operații eficiente de reuniune și identificare.	*	+	+	+

Subiecte	Descriere succintă *Structuri de date*	7-9	10	11	12
6. Arbori binari (noțiuni, parcurgeri)	Structuri arborescente utilizate pentru modelarea relațiilor ierarhice și parcurgeri sistematice.	*	+	+	+
7. Segment Tree (operații de bază)	Structură pentru procesarea eficientă a interogărilor și actualizărilor pe intervale.	-	*	+	+
8. Fenwick Tree (Binary Indexed Tree)	Structură optimizată pentru calculul sumelor prefix și actualizări punctuale.	-	*	*	+
9. Lazy propagation	Extensie a arborilor de intervale pentru gestionarea eficientă a actualizărilor pe intervale.	-	-	*	*

C. Șiruri de caractere

Această secțiune abordează prelucrarea șirurilor de caractere ca domeniu distinct, punând accent pe tehnici algoritmice specifice și pe utilizarea programării dinamice în rezolvarea problemelor clasice.

Subiecte	Descriere succintă *Șiruri de caractere*	7-9	10	11	12
1. Prelucrări simple pe șiruri	Operații elementare asupra șirurilor de caractere, precum accesarea, modificarea și compararea secvențelor de caractere.	+	+	+	+
2. Căutare naivă	Determinarea apariției unui șablon într-un șir, prin comparări directe, fără tehnici de optimizare.	+	+	+	+
3. Subșir comun	Determinarea unui subșir comun maxim între două șiruri, utilizând programare dinamică clasică.	-	+	+	+

Subiecte	Descriere succintă *Șiruri de caractere*	7-9	10	11	12
4. Distanța de redactare	Calculul numărului minim de operații necesare pentru transformarea unui șir în altul, folosind programare dinamică (Edit Distance – DP).	-	-	*	*
5. Căutare eficientă în șiruri	Tehnici de determinare a apariției unui șablon într-un șir, utilizând informații preprocesate pentru reducerea numărului de comparații.	-	-	*	*

D. Algoritmi și tehnici de programare

Această secțiune urmărește dezvoltarea gândirii algoritmice prin studiul tehnicilor fundamentale de rezolvare a problemelor, de la algoritmi de bază la tehnici de programare precum greedy, divide et impera și programarea dinamică.

Subiecte	Descriere succintă *Algoritmi și strategii algoritmice*	7–9	10	11	12
1. Căutare liniară și binară	Tehnici de căutare a unui element într-o colecție de date, prin parcurgere secvențială sau prin împărțirea repetată a domeniului de căutare.	+	+	+	+
2. Sortări elementare	Algoritmi de sortare simpli, bazați pe comparații directe și schimburi succesive de elemente.	+	+	+	+
3. Sortări eficiente (subcuadratice)	Algoritmi de sortare cu complexitate subcuadratică, utilizați pentru volume mari de date.	*	+	+	+
4. Tehnica Greedy	Strategii de rezolvare care construiesc soluția pas cu pas, alegând local opțiunea optimă.	*	+	+	+
5. Metoda desparte şi stăpâneşte (tehnica divide et impera)	Strategii algoritmice bazate pe descompunerea problemei în subprobleme independente și combinarea soluțiilor.	*	+	+	+
6. Metoda reluării (tehnica backtracking)	Explorarea sistematică a spațiului soluțiilor prin construcții parțiale și reveniri controlate.	*	+	+	+
7. Programare dinamică – concepte generale (DP)	Modelarea problemelor prin stări și tranziții, cu memorarea rezultatelor intermediare pentru evitarea recalculării.	*	+	+	+

Subiecte	Descriere succintă *Algoritmi și strategii algoritmice*	7–9	10	11	12
8. DP pe grafuri și arbori	Aplicarea programării dinamice pe structuri de tip graf sau arbore, în contexte de optimizare sau numărare.	-	-	+	+
9. DP cu bitmask	Reprezentarea stărilor prin valori binare, pentru rezolvarea problemelor cu spațiu de soluții limitat, utilizând programare dinamică.	-	-	*	+
10. Optimizări ale programării dinamice	Tehnici generale de reducere a complexității spațiale sau temporale în soluțiile bazate pe DP.	-	-	-	*

E. Grafuri

Această secțiune urmărește înțelegerea grafurilor ca structuri de date complexe și modele simple pentru reprezentarea relațiilor dintre obiecte și rezolvarea problemelor de parcurgere, conexiune și optimizare, întâlnite frecvent în informatică.

Conținut	Descriere succintă *Grafuri*	7–9	10	11	12
1. Noțiuni de bază. BFS și DFS	Reprezentarea grafurilor și parcurgerea acestora în lățime și în adâncime, inclusiv prin aplicații pe tablouri bidimensionale și alte structuri discrete, pentru explorare și determinarea proprietăților de bază.	*	+	+	+
2. Grafuri orientate și grafuri aciclice (DAG)	Grafuri cu muchii orientate și proprietăți specifice grafurilor fără cicluri.	-	+	+	+
3. Sortare topologică	Determinarea unei ordonări a nodurilor într-un graf aciclic orientat, respectând relațiile de precedență.	-	+	+	+

Conținut	Descriere succintă *Grafuri*	7–9	10	11	12
4. Drumuri minime	Determinarea drumurilor de cost minim între noduri, în grafuri ponderate sau neponderate.	-	+	+	+
5. Componente tare conexe și biconexitate	Analiza structurii grafurilor prin identificarea componentelor cu proprietăți de conectivitate.	-	+	+	+
6. Arbori parțiali de cost minim	Construirea unui subgraf de tip arbore care conectează toate nodurile cu cost total minim.	-	*	+	+
7. Probleme speciale pe grafuri	Identificarea grafurilor bipartite și rezolvarea problemelor de potrivire și repartizare între două mulțimi de noduri, inclusiv prin modelarea acestora cu ajutorul rețelelor de flux.	-	-	*	+

F. Teoria mulțimilor și combinatorică

Această secțiune urmărește dezvoltarea gândirii logice și a capacității de numărare prin studierea relațiilor dintre elemente, a modurilor de combinare și a problemelor de numărare întâlnite frecvent în informatică.

Conținut	Descriere succintă	7–9	10	11	12
1. Algebra booleană	Operații logice fundamentale și proprietăți ale expresiilor booleene, utilizate în modelarea condițiilor și a deciziilor.	+	+	+	+
2. Permutări, combinări și aranjamente	Probleme de numărare bazate pe ordonarea sau selecția elementelor, cu sau fără restricții.	+	+	+	+

3. Plasări	Numărarea aranjamentelor de elemente selectate dintr-un set, ținând cont de ordine și de condiții impuse.	-	+	+	+
4. Principiul incluziunii–excluziunii	Metodă de numărare pentru situații în care este necesară corectarea suprapunerilor dintre mai multe mulțimi sau condiții	-	+	+	+
5. Numere speciale definite recursiv	Definirea și utilizarea relațiilor de recurență simple, cu aplicații în numărare și programare dinamică (de exemplu Fibonacci, Catalan, Bell).	*	*	+	+

G. Geometrie computațională

Această secțiune urmărește aplicarea noțiunilor geometrice de bază în rezolvarea problemelor algoritmice, prin utilizarea coordonatelor, a distanțelor și a relațiilor geometrice simple.

Conținut	**Descriere succintă** *Geometrie algoritmică*	**7–9**	**10**	**11**	**12**
1. Noțiuni fundamentale și coordonate	Reprezentarea punctelor în plan, sisteme de coordonate și relații geometrice elementare.	+	+	+	+
2. Distanțe și arii elementare	Calculul distanțelor dintre puncte și al ariilor figurilor geometrice simple.	+	+	+	+
3. Reprezentare vectorială	Utilizarea vectorilor pentru descrierea pozițiilor și deplasărilor în plan.	-	+	+	+
4. Intersecții simple și orientare	Determinarea relațiilor geometrice dintre segmente și puncte, inclusiv orientarea relativă.	-	+	+	+

Conținut	Descriere succintă *Geometrie algoritmică*	7–9	10	11	12
5. Aria poligonului simplu	Calculul ariei unui poligon simplu definit prin coordonatele vârfurilor.	-	-	+	+
6. Convex Hull	Determinarea învelișului convex al unui set de puncte în plan.	-	-	-	*
7. Optimizări geometrice	Probleme geometrice ce implică optimizarea unor mărimi, fără utilizarea tehnicilor avansate de calcul numeric.	-	-	-	*

H. Aspecte teoretice

Această secțiune urmărește formarea abilităților de cunoaștere și utilizare a conceptelor fundamentale ale Informaticii Teoretice pentru rezolvarea eficientă a problemelor algoritmice, prin utilizarea tipurilor și structurilor de date adecvate, aprecierea complexității algoritmilor implementați și a resurselor de memorie disponibile.

Conținut	Descriere succintă *Aspecte teoretice*	7–9	10	11	12
1. Limitări de reprezentare a numerelor și caracterelor	Limitări de utilizare a numerelor întregi și reale în funcție de tipul de date folosit. Codificări ASCII și ASCII extins	+	+	+	+
2. Operații elementare	Calcularea estimativă a numărului de operații într-o expresie / instrucțiune, fragment de cod, funcție, etc.	-	+	+	+
3. Elemente de complexitate asimptotică	Formarea funcției (poligonului sau expresiei exponențiale) pentru numărarea operațiilor elementare din algoritm, identificarea	-	-	*	+

Conținut	Descriere succintă *Aspecte teoretice*	7–9	10	11	12
	factorului dominant, stabilirea funcției de complexitate				
4. Metaoperații	Identificarea complexității operațiilor standard în structurile de date: adăugare / lichidare element, căutare element, căutare maximal, minimal, cu proprietăți date.	-	-	*	+
5. Clase de probleme	Identificarea abordărilor și strategiilor corecte pentru rezolvarea informatică a problemelor. Capacitatea de selectare a strategiei de rezolvare în funcție de specificul subtask-urilor propuse	-	-	-	*

Structura probelor de concurs

Proba de concurs la disciplina Informatică constă din probleme de natură algoritmică, formulate într-un format unitar, care permite înțelegerea clară a cerinței, implementarea soluției și evaluarea automată a programelor elaborate de competitori.

Fiecare problemă este prezentată într-o formă standardizată și include, în mod obligatoriu, următoarele componente:

- Titlul problemei;
- Descrierea / contextul problemei, cu formularea explicită a sarcinii;
- Datele de intrare, cu specificarea exactă a formatului acestora;
- Datele de ieșire, cu specificarea exactă a rezultatului ce trebuie afișat;
- Restricțiile problemei, care pot viza dimensiunea datelor, timpul de execuție, memoria disponibilă sau interdicții explicite;
- Exemplu(e) de intrare–ieșire, cu rol ilustrativ.

Această structură este utilizată în mod consecvent pentru toate problemele din proba de concurs, indiferent de nivelul de dificultate sau etapa olimpiadei.

În continuare este prezentat un exemplu de problemă, formulată conform structurii descrise mai sus.

Exemplu de problemă

Suma numerelor mari

Timp: 1 secundă Memorie: 64 MB

Descriere

Se dau două numere naturale foarte mari A și B. Dimensiunea acestora poate depăși capacitatea tipurilor numerice standard.

Sarcină: Elaborați un program care determină suma numerelor A și B.

Date de intrare. Intrarea standard conține, pe o singură linie, două numere naturale A și B, separate printr-un spațiu.

Date de ieșire. Ieșirea standard va conține un singur număr natural, suma numerelor A și B.

Restricții. $10^{10} \leq A, B \leq 10^{100}$.

Exemplu.

Intrare	*Ieșire*
`1234567890 987654321011`	`988888888901`

Soluția problemei

Numerele date pot avea un număr foarte mare de cifre și, prin urmare, nu pot fi memorate în tipuri numerice standard. Soluția constă în prelucrarea numerelor ca șiruri de caractere și simularea adunării cifră cu cifră, începând de la ultima cifră (dreapta), similar metodei clasice de adunare manuală.

Algoritmul parcurge fiecare cifră o singură dată. La fiecare pas:

1. se adună cifrele corespunzătoare și transportul (`transport`);
2. cifra rezultată este `sum % 10`;
3. transportul pentru pasul următor este `sum / 10`.

Complexitate:

- timp: `O(n)`, *unde* `n = max(`$|A|,|B|$`);`
- memorie: `O(n)`.

Implementare **C++** [2]

```cpp
#include <iostream>

using namespace std;

#define n 102 // dimensiunea structurilor va depasi limitele impuse

int lun(char x[]) // functia calculeaza lungimea unui sir de caractere
{
    int i = 0;
    while (x[i] != '\0') i++;
    return i;
}

void ajustare (char x[]) // se formeaza numarul pe cifre, aliniere - dreapta
{
    int h = lun (x);
    for (int i = 1; i <= h; i++) x[n - i] = x[h - i];
    for (int i = 0; i < n - h; i++) x[i] = '0';
    return;
}

int main()
{
    // declarare structuri de date
    char sA[n], sB[n];
    int  sSum[n], transport = 0, h = 0;
```

[2] În exemplele de soluții, s-a preferat implementări bazate pe operații elementare (indexare și parcurgere), pentru a evita dependența de construcții avansate ale bibliotecilor.

```
    // citire date. sirul de caractere se aliniaza dreapta
    cin >> sA >> sB;
    ajustare(sA);
    ajustare(sB);

    // calcul suma, folosind regulile clasice ale aritmeticii
    for (int i = n - 1; i >= 0; i--)
    {
        sSum[i] = sA[i] -'0' + sB[i] -'0' + transport;
        transport = sSum[i] / 10;
        sSum[i] = sSum[i] % 10;
    }
    // afisare rezultat cu eliminarea prefixului de zero-uri
    while (!sSum[h]) h++;
    for (int i = h; i < n; i++) cout << sSum[i];

    return 0;
}
```

Implementare Pascal

```
Program SumaNumerelorMari;
uses SysUtils;
var
  A, B, R, line: string;
  iA, iB, s, transport, da, db, p: integer;

begin
  readln(line);

  p := Pos(' ', line);
  A := Copy(line, 1, p - 1);
  B := Copy(line, p + 1, Length(line));

  if Length(A) < Length(B) then
  begin
    R := A; A := B; B := R;
  end;
  R := '';

  carry := 0;
  iA := Length(A);
  iB := Length(B);
```

```
  while iA > 0 do
  begin
    da := Ord(A[iA]) - Ord('0');
    if iB > 0 then db := Ord(B[iB]) - Ord('0')
    else db := 0;

    s := da + db + transport;
    R := Chr(Ord('0') + (s mod 10)) + R;
    transport := s div 10;

    Dec(iA);
    Dec(iB);
  end;

  if transport> 0 then
    R := Chr(Ord('0') + carry) + R;

  writeln(R);
end.
```

Criterii de apreciere

1. Evaluarea probelor de concurs se realizează în conformitate cu baremele de verificare transmise de către ANCE și Consiliul Olimpic Republican.
2. Baremul de verificare cuprinde, pentru fiecare problemă inclusă în proba de concurs, un set de teste, utilizate pentru evaluarea soluțiilor prezentate de competitori.
3. Aplicarea baremului de verificare se realizează în conformitate cu următoarea procedură:
    a. Programul elaborat de competitor este rulat pentru fiecare test aferent problemei, iar datele de ieșire generate sunt comparate cu datele corespunzătoare soluției problemei;
    b. Un test este promovat în cazul în care soluția furnizată de programul evaluat este corectă conform mecanismului de evaluare stabilit pentru problema respectivă, respectând simultan restricțiile privind timpul de execuție și volumul de memorie, prevăzute în enunțul problemei;

        Notă: Mecanismul de evaluare poate presupune compararea cu date de ieșire de referință, evaluare prin program verificator (checker), sau alte proceduri specifice, definite în documentația problemei.

    c. Numărul de puncte aferent fiecărui test este stabilit în fișa „Descrierea generală a problemelor”, care face parte integrantă din documentația probei scrise desfășurate la calculator. Pentru testele nepromovate se acordă zero puncte;
    d. Punctajul obținut de competitor pentru fiecare problemă este determinat ca fiind cel mai mare punctaj obținut de acesta în urma evaluării tuturor submisiilor aferente problemei respective. Punctajul total al competitorului se calculează prin însumarea punctajelor astfel determinate pentru toate problemele probei de concurs;
    e. Clasamentul competitorilor se stabilește în ordinea descrescătoare a punctajelor totale obținute;
    f. În situația în care doi sau mai mulți competitori obțin același punctaj total, departajarea acestora se realizează în funcție de momentul în care fiecare competitor a atins pentru prima dată punctajul final total, determinat conform prevederilor literei d). Competitorul care a atins mai devreme acest punctaj este clasat pe o poziție superioară;

        Notă. Momentul atingerii punctajului final total reprezintă momentul în care suma celor mai bune punctaje obținute de competitor la toate problemele probei de concurs a atins pentru prima dată valoarea punctajului total final al acestuia. Submisiile efectuate ulterior acestui moment nu influențează criteriul de departajare.