DIDACTIC APPROACHES IN PERFORMANCE INSTRUCTION

ABORDĂRI DIDACTICE ÎN INSTRUIREA DE PERFORMANȚĂ

GLOBA Angela, PhD, Associate Professor,
"Ion Creangă" State Pedagogical University of Chisinau
ORCID ID: 0000-0002-2653-0320

CZU: 374.1:004
DOI: 10.46727/c.27-28-09-2024.p342-350

Abstract. *A problem for a computer science olympiad has a long history and continues to be relevant for many generations of children interested in computer science. This article examines the didactic aspects of creating a competition problem. Proposed examples come to elucidate the importance of each stage of problem creation, the psychological aspects related to competition problems, etc.*

Keywords: *computer science olympiad, competition problem, problem solution, algorithm, algorithm efficiency, programming techniques.*

Introducere

Olimpiadele școlare sunt o formă specială a activităților extracurriculare. Importanța olimpiadelor de Informatică nu mai necesită demonstrații. Acest fapt de datorează digitalizării continue a activităților profesionale și cotidiene. În general, acest lucru este direct proporțional cu rolul informaticii în societatea modernă. Esența unei olimpiade de informatică sunt problemele propuse competitorilor. În dependență de complexitatea și frumusețea problemei, acestea devin istorie motivând încă multe generații de copii dotați și supradotați la informatică. Elaborarea unei probleme pentru o olimpiadă de informatică este un proces greu, minuțios, dar, în același timp, foarte creativ [1,2,3]. Tendința modernă de a formula problemele în limbaj natural este extrem de importantă, dar creează dificultăți pentru autori (subiectul trebuie să fie credibil).

Un alt aspect, mai puțin abordat, este corectitudinea problemei. Pentru a exemplifica despre ce este vorba voi aduce un exemplu: În filmul „Mathematician in Wonderland" (Coreea de Sud, 2022) este un episod unde Lee Hak-Seong un geniu al matematicii care a evadat din Coreea de Nord pentru aș găsi libertatea academică, și ascunzându-și identitatea sub paznicul școlii în cadrul primei ore de matematică cu Han Ji-Woo, un elev dintr-un liceu privat prestigios, care nu se înscrie în acest liceu din cauza diferențelor sociale cu colegii săi înstăriți și este pe punctul de a fi exmatriculat din cauza problemelor la matematică, îi propune să rezolve o problemă simplă de geometrie: Într-un triunghi dreptunghic înălțimea $|CO| = 6\ cm$, iar $|AB| = 10\ cm$. Să se calculeze aria triunghiului (figura 1).

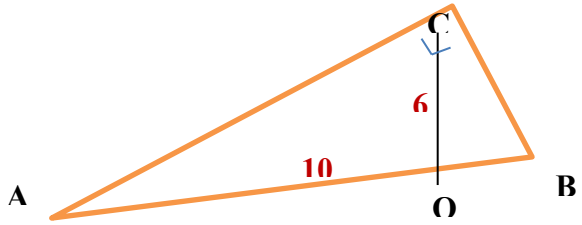


Figura 1. Triunghi dreptunghic cu înălțimea $|CO| = 6\ cm$ și ipotenuza $|AB| = 10\ cm$

Han Ji-Woo, și noi toți, începem prin a scrie formula $A = \frac{1}{2}|CO| \cdot |AB|$. De unde se obține că $A = 30\ cm^2$. Când profesorul Lee Hak-Seong a afirmat că răspunsul este greșit, reacția a fost de a face calculul din nou. Apoi a urmat desenul din figura 2, de unde este foarte clar că așa triunghi nu există (AB – diametrul cercului circumscris triunghiului dreptunghic; ipotenuza este mai mică decât cateta).

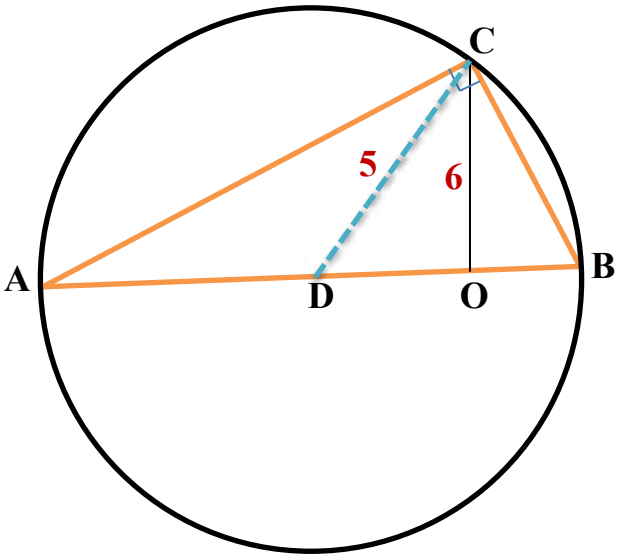


Figura 2. Cerc circumscris triunghiului dreptunghic

Concluzia care urmează, ținând cont de exemplul de mai sus, este că, de regulă, omul, în special elevii, se concentrează pe rezolvarea problemei și mai puțin pe corectitudinea problemei, în particular a datelor. Cu atât mai mult, copiii dotați și supradotați nu acceptă sau acceptă cu greu ca o problemă să fie formulată incorect, adică să conțină erori în formulare sau în datele de intrare [4,5]. Acest fapt confirmă suplimentar că procesul de elaborare a unei probleme necesită, în primul rând, specialiști cu calități profesionale înalte și, în al doilea rând, cunoașterea metodologiei de crearea a problemelor.

Model de creare a unei probleme de concurs

Elaborarea problemelor de concurs este o sarcină destul de dificilă pentru autori, motiv pentru care necesită abordări metodice complexe, care țin cont de: pregătirea teoretică și practică a participanților; aspectul psihic și situațiile de stres; complexitatea concursului etc. Setul minim de resurse pentru o problemă de concurs este indicat în figura 3 [6].

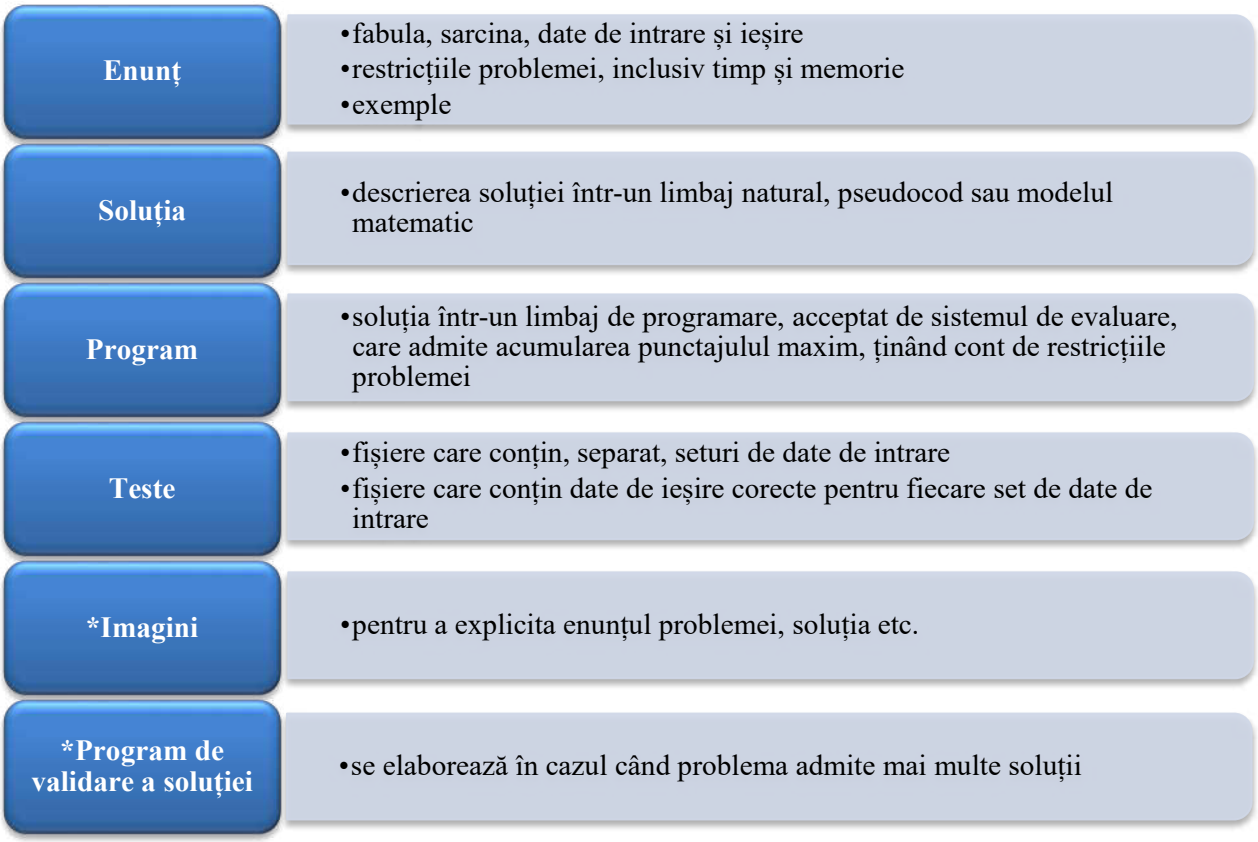


Figura 3. Setul minim de resurse pentru o problemă de concurs

Prezentăm mai jos un model de creare a unei probleme pentru olimpiada de informatică.

Pasul* 1. *Crearea enunțului. Fabula.

1.1. Se alege un domeniu (matematică, biologie, chimie etc.) sau se merge pe ideea utilizării unor structuri de date sofisticate, sau ambele [7].

De exemplu: Fie dată matricea A[1..*n*,1..*m*]. Cu fiecare număr din prima linie se pot forma sume de numere naturale în felul următor:

• se porneşte cu un numărul din linia 1;

• succesorul lui se află pe linia următoare plasat sub el sau pe diagonală la dreapta sau pe diagonală la stânga. Care este cea mai mare suma care se poate forma astfel şi care sunt numerele care o alcătuiesc?

	1	2	3	4
1	7	3	5	1
2	7	3	6	8
3	9	1	6	2
4	1	3	0	9
5	9	4	9	2

1.2. Se adaugă condiții suplimentare.

De exemplu: dacă toți succesorii posibili din linia *i*+1 sunt egali cu 0, atunci deplasarea de la linia *i* la linia *i*+1 nu poate fi efectuată, iar suma poate fi formată doar cu numerele din liniile 1..*i*; dacă deplasarea de la o linie la alta solicită un cost, atunci vom căuta costul minim; etc.

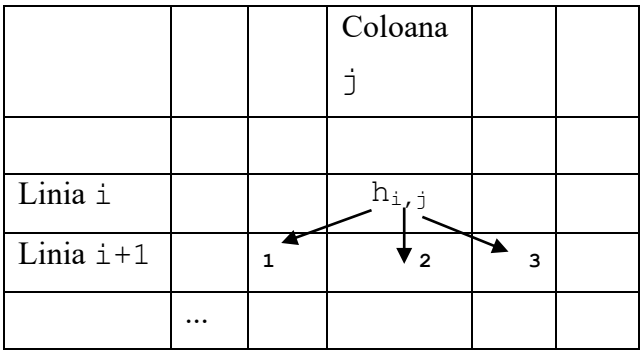
	1	2	3	4
1	7	3	5	1
2	7	3	6	8
3	9	1	6	2
4	1	0	0	0
5	9	4	9	2

1.3. Crearea fabulei;

De exemplu: ***Practica de specialitate*** (*Olimpiada Republicană la Informatică 2023*).

Universitatea pregătește specialiști la, preconizează să trimită la practică pe acel litoral un număr maxim de studenți respectând următoarele condiții:

- în mod obligatoriu va fi ales un hotel din prima linie;
- hotelurile alese reprezintă o rețea formată dintr-un șir de linii programul de studii *Servicii hoteliere, turism şi agrement* și intenționează să trimită studenții, pentru a-și desfășura stagiul de practica de specialitate, pe litoralul mării Mediterane, în apropiere de orașul Antalya, unde sunt mai multe hoteluri Ultra All Inclusiv. Hotelurile sunt aranjate în `n` linii și `m` coloane. Pentru fiecare student, universitatea achită o sumă `S` care depinde de linia în care se află hotelul. Un hotel poate asigura practica de specialitate pentru `x` studenți. Se știe că sunt hoteluri care nu primesc studenți la practica de specialitate. Decanul facultății consecutive alese astfel: dacă a fost ales hotelul $h_{i,j}$ din linia `i`, atunci din linia `i+1` se poate alege un hotel care se află sau pe diagonală la stânga sau vertical în jos sau pe diagonală la dreapta (vezi figura și ordinea alegerii hotelurilor (1,2,3)) numai în cazul când hotelurile din linia `i+1` și coloanele `j-1,j,j+1` primesc studenți;



Pasul 2. Crearea sarcinii.

De exemplu: **Sarcină:** Elaborați un program care ar calcula numărul maxim de studenți pe care Universitatea poate să-i trimită pentru a-și desfășura stagiul de practica de specialitate pe litoralul mării Mediterane, suma totală minimă pe care trebuie s-o achite Universitatea și rețeaua de hoteluri în care își vor desfășura practica studenții.

Pasul 3. Definirea datelor de intrare/ieșire.

3.1. Stabilirea datelor de intrare, tipul lor (care poate să nu fie atât de explicit) și metoda de citire a acestora;

De exemplu: **Date de intrare:** Prima linie a intrării standard conține două numere naturale despărțite printr-un spațiu: *n* (numărul de linii) și `m` (numărul de coloane). Următoarele `n` linii vor conține câte `m` numere naturale pe fiecare linie separate printr-un spațiu – numărul `x` de studenți pe care îi poate primi hotelul pentru a-și desfășura practica de specialitate (pentru hotelurile care nu primesc studenți la practică de specialitate se indică 0). Linia `n+2` va conține `n` numere naturale – suma pe care o achită universitatea pentru un student începând cu linia de hoteluri 1.

Observație: În cazul când citirea se face de la intrarea standard, atunci se va scrie ”Intrarea standard va conține...”

3.2. Stabilirea datelor de ieșire, tipul lor (care poate să nu fie atât de explicit) și metoda de citire a acestora;

De exemplu: **Date de ieșire:** Ieșirea standard va conține pe prima linie un număr natural – numărul maxim de studenți pe care decanul îi poate trimite pe litoralul mării Mediterane pentru a desfășura practica de specialitate. Pe linia a doua se va afișa suma totală minimă pe care trebuie s-o achite Universitatea pentru practica studenților. Începând cu linia a treia, se vor afișa coordonatele hotelurilor începând cu hotelul din prima linie (numărul liniei, numărul coloanei separate printr-un spațiu). Se va ține cont de următoarele: dacă suma totală minimă pentru numărul maxim de studenți se poate obține în mai multe moduri, atunci se va afișa soluția pentru care numărul de ordine a coloanei în care se află primul hotel este cel mai mic.

Observație: în cazul când afișarea datelor se face într-un fișier, atunci se va scrie ”Fișierul `hotel.out` va conține...”

***Pasul* 4. *Stabilirea restricțiilor pentru datelor de intrare/ieșire și a celor de complexitate* (*timp, memorie*).**

De exemplu: **Restricții:** $4 \leq n,m \leq 100$, $0 \leq x \leq 100$, $1 \leq S \leq 250$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `hotel.pas`, `hotel.c` sau `hotel.cpp`.

La stabilirea restricțiilor se va ține cont de algoritmul de rezolvare, structurile de date utilizate etc. De exemplu, o metodă de rezolvare presupune utilizarea a trei matrici (I – păstrăm datele inițiale; II – pentru calcularea numărului maxim de studenți; III – pentru memorarea coordonatelor rețelelor posibile de hoteluri), celelalte date de intrare sunt nesemnificative ca volum de memorie. De asemenea, autorul problemei va ține cont ca la testarea problemei pe setul de teste, timpul de calcul nu trebuie să fie mai mare decât 3-5 secunde, apoi va transmite problema pentru a fi testată pe evaluator. După testarea problemei pe evaluator se vor stabili restricțiile de timp și memorie care vor fi comunicate competitorilor.

Restricțiile pot fi stabilite și pentru fiecare subtask aparte (problema *Rectangles*, IOI, 2019, [8]):

Subtasks	**Punctaj**	**Restricții**
1 11 tests	8	$n, m \leq 30$
2 8 tests	7	$n, m \leq 80$
3 8 tests	12	$n, m \leq 200$
4 16 tests	22	$n, m \leq 700$
5 11 tests	10	$n \leq 3$
6 8 tests	13	$0 \leq a[i,j] \leq 1$(for all $0 \leq i \leq n-1, 0 \leq j \leq m-1$)
7 18 tests	28	No additional constraints
Total	**100** puncte	

Observație: De regulă acest lucru este făcut la Olimpiada Republicană de Informatică din Republica Moldova, olimpiadele internaționale sau restricțiile sunt unice, iar bateriile de teste sunt create pentru a lua în calcul algoritmul implementat.

Pasul* 5. *Crearea exemplelor care ilustrează un set de date de intrare* (*sau mai multe*) *și datele de ieșire obținute.

Exemplele se aduc, în mare parte, pentru a clarifica citirea și afișarea datelor. În același timp, exemplele propuse nu trebuie, în nici un caz, să sugereze algoritmul de rezolvare (din contra!). Autorul nu este obligat să aducă exemple pentru a elucida toate cazurile posibile.

Exemple:

Intrare	*Ieşire*
4 6 9 21 8 41 5 24 20 0 0 0 1 18 0 1 0 0 0 0 0 0 0 5 5 5 100 75 50 25	42 3650 1 2 2 1 3 2
5 5 0 0 2 0 0 0 3 1 3 0 0 0 2 0 0 0 0 3 0 0 0 4 0 4 0 100 75 50 25 10	14 640 1 3 2 2 3 3 4 3 5 2

Explicații: Pentru primul exemplu, numărul maxim de studenți se poate obține în trei moduri alegând hotelurile: (a) (1,2), (2,1), (3,2) cu suma totală 3650=21*100+20*75+1*50; b) (1,4), (2,5) cu suma totală 4175=41*100+75; (c) (1,6), (2,6) cu suma totală 3750=24*100+18*75. Așa cum, suma totală minimă este 3650, se va alege rețeaua de hoteluri (a). Se poate observa că din hotelurile (1,3), (3,2), (2,5), (2,6) nu se poate efectua deplasarea spre linia următoare.

Pentru exemplul 2, numărul maxim de studenți se poate obține în următoarele moduri alegând hotelurile:

a) (1,3), (2,2), (3,3), (4,3), (5,2); b) (1,3), (2,2), (3,3), (4,3), (5,4);

c) (1,3), (2,4), (3,3), (4,3), (5,2); d) (1,3), (2,4), (3,3), (4,3), (5,4).

Ținând cont de ordinea alegerii hotelurilor (1,2,3, vezi figura) se va afișa varianta a).

Observație: De regulă, exemplele sunt însoțite de explicații, însă - nu este obligatoriu.

Pasul* 6. *Prezentarea soluției.

Este vorba despre descrierea soluției într-un limbaj natural, pseudocod sau modelul matematic. *De exemplu*: Dacă se va încerca să se construiască toate variantele posibile de rețele de hoteluri, atunci complexitatea algoritmului creat va fi exponențială. În acest caz, este bine de utilizat tehnica programării dinamice [9]. Vom analiza şirul (rețeaua) cu maxim *n* numere (care respectă condițiile problemei şi care formează numărul maxim de studenți). Vom analiza în acest șir hotelul ales din linia *i*. Hotelurile alese între liniile *i+1* şi *n*, formează un număr maxim de studenți în raport cu numărul de studenți care se poate obține începând cu hotelul ales din linia

i, în caz contrar nu se vor respecta condițiile problemei. Așa dar, fie că am memorizat hotelurile într-o matrice cu *n* linii și *m* coloane – `Loc[i,j]`. Vom crea matricea `Aux[i,j]` începând cu linia *n* spre linia 1, unde pe linia *i* sunt memorate numărul maxim de studenți care se poate obține începând cu linia *i* spre linia *n*. Astfel, obținem:

$$Aux[i,j] = \begin{cases} Loc[i,j], i = n, j = \overline{1,m} \\ \max\{Loc[i,j] + Aux[i+1,j-1]; Loc[i,j] + Aux[i+1,j]; Loc[i,j] + Aux[i+1,j+1]\}, \\ i = \overline{1,n-1}, j = \overline{1,m} \end{cases}$$

În linia $Aux[1,j]$, $j = \overline{1,m}$ vom avea numărul maxim de studenți care pot fi trimiși la practică pentru rețeaua de hoteluri care începe cu un hotel din prima linie și coloana *j*.

Deoarece, din condițiile problemei, este necesar de a afișa rețeaua de hoteluri alese, o vom memora într-o matrice cu *n* linii și *m* coloane – `retea[i,j]`, unde pentru fiecare $i = \overline{1,n-1}, j = \overline{1,m}$ se va memoriza numărul coloanei din care se alege succesorul lui `Loc[i,j]`.

În vectorul `pret[i]`, $i = \overline{1,n}$, se memorează prețul achitat de universitate pentru fiecare student care face practica într-un hotel din linia *i*.

În matricea `Nr_max` cu 3 linii și *m* coloane se va memora:

- `Nr_max[1,j]`: numărul coloanei din prima linie de hoteluri cu care începe rețeaua de hoteluri pentru care se obține un număr maxim de studenți;
- `Nr_max[2,j]`: numărul de linii din rețeaua de hoteluri care începe cu hotelul din colona *j* pentru care se obține un număr maxim de studenți;
- `Nr_max[3,j]`: suma totală care se va achita de universitate dacă se alege rețeaua de hoteluri care începe cu hotelul din colona *j* pentru care se obține un număr maxim de studenți.

Loc	1	2	3	4	5	6
1	9	21	8	41	5	24
2	20	0	0	0	1	18
3	0	1	0	0	0	0
4	0	0	0	5	5	5

Aux	1	2	3	4	5	6
1	30	42	8	42	23	42
2	21	0	0	0	1	18
3	0	1	0	0	0	0
4	0	0	0	5	5	5

retea	1	2	3	4	5	6
1	1	1	0	5	6	6
2	2	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0

Pasul 7. Scrierea codului într-un limbaj de programare.

Acest pas presupune prezentarea soluției într-un limbaj de programare acceptat de sistemul de evaluare care admite acumularea punctajului maxim, ținând cont de restricțiile problemei. De asemenea, în cazul când problema admite mai multe soluții se va elabora un program de validare a soluției.

Pasul 8. Elaborarea testelor de evaluare a soluției. programare.

La acest pas se creează fișierele `*.in` și `*.out` cu scopul de a evalua algoritmii propuși de competitori în funcție de tehnica de programare aleasă sau structurile de date aplicate. Se vor crea următoarele tipuri de teste: teste la limită (dimensiuni minime/maxime); teste pentru cazuri particulare; teste pentru exactitatea calculelor cu numere reale; teste care scot în evidență caracteristicile specifice ale limbajului sau mediului de programare; teste care țin cont de tehnica de programare implementată.

Observație: Numărul de teste pentru o problemă este aleatoriu, dar nu mai puțin de 10. În funcție de concurs, exemplul poate fi inclus într-un test. Testele sunt grupate în funcție de tehnica de programare aleasă.

Tehnici de programare	Descriere	Punctaj
Brute force, backtracking	Dacă se va încerca să se construiască toate variantele posibile de rețele de hoteluri, atunci complexitatea algoritmului creat va fi exponențială. $4 \le n,m \le 10,\ 0 \le x \le 100,\ 1 \le S \le 250$	10
Structuri dinamice de date	Soluția posibilă generată (o stivă) se memorează într-un vector, apoi, se generează o altă soluție posibilă; se calculează numărul maxim de studenți și se memorează acea soluție care are suma maximă; același lucru se face pentru costul minim și rețea; timpul de calcul va fi excesiv; $4 \le n,m \le 20,\ 0 \le x \le 100,\ 1 \le S \le 250$	30
Programare Dinamică	Implementarea tehnicii Programării Dinamice admite o complexitate temporală egală cu ($O(n^2)$). $4 \le n,m \le 100,\ 0 \le x \le 100,\ 1 \le S \le 250$	60

Pasul 10. Analiza problemei de către Comitetul Olimpic.

Comitetul Olimpic analizează problema din toate punctele de vedere (complexitate, domeniu, grad de dificultate etc.). Dacă este cazul, se fac modificări. Ulterior, problema se încarcă pe server, apoi se verifică bateriile de teste, se clarifică restricțiile temporale și de memorie. Se califică problema ca probă de concurs. În final, problema se traduce.

Concluzii

Pentru elaborarea unei probleme de concurs autorul are nevoie de o pregătire profesională foarte înaltă. Enunțul trebuie să fie clar, fără a admite ambiguități. Este importantă fiecare virgulă. Evident, verificarea corectitudinii algoritmului creat este primordială.

Corectitudinea seturilor de teste este esențială în organizarea și desfășurarea calitativă a unui concurs de programare. Setul de teste creat pentru o problemă trebuie verificat minuțios prin aplicarea diverselor softuri (dacă este posibil), fie manual. De regulă, se demonstrează corectitudinea algoritmului elaborat.

Pregătirea psihologică este esențială pentru actorii (elevi, profesori) care se implică într-o competiție. Sarcina profesorilor este de a forma la elevi o atitudine pozitivă atât față de ceilalți concurenți cât și față de juriu.

Sunt necesare investiții majore în dezvoltarea instruirii de performanță la informatică atât pentru profesori cât și pentru elevi.

Bibliografie:

1. Skiena S., Revilla M. Programming challenges. The Programming contest Training Manual. Springer, 2002.
2. Оршанский С. А. О решении олимпиадных задач по программированию формата ACM ICPC. В: Журнал „Мир ПК”, Nr. 9, 2005.
3. Globa A., Gasnaș A. Metodologia elaborării problemelor de concurs. În: Mathematics & Information Technologies: Research and Education (MITRE-2023), International Conference. Satellite conference of the 10th Congress of Romanian Mathematicians, Chișinău, June 26-29, 2023. Abstract. Chișinău, CEP USM, 2023, p. 102.
4. Chiriac L., Globa A., ș. a. Performanțele academice și activitatea profesională ale elevilor dotați și supradotați. Implementarea conceptului de inter/transdisciplinaritate în învățământul preuniversitar. Capitole în studiul monografic: L. Chiriac ș.a. Evaluarea procesului de studiere a științelor reale și ale naturii din perspectiva inter/transdisciplinarității. Concept STEAM, Chișinău, Tipografia Centrală, 2020, 252 p. ISBN 978-9975-117-50-0.
5. Globa A., Corlat S., Gasnaș A. Opțiuni de continuare a studiilor pentru elevii olimpici din Republica Moldova la disciplina informatica. În: Acta et Commentationes, Științe ale Educației. Revistă științifică Nr.3(21), 2020. Chișinău: Universitatea de Stat din Tiraspol, 2020. p. 23-31. ISSN 1857-0623, E-ISSN 2587-3636.
6. Globa A., Corlat S. Didactic aspects regarding to creating test sets for competition problems. În: Acta et Commentationes. Științe ale Educației. Revistă științifică Nr.4(18) (2019). Chișinău: Universitatea de Stat din Tiraspol, 2019. p. 73-85. ISSN 1857-0623, E-ISSN 2587-3636.
7. Braicov A., Globa A. Soluții originale pentru probleme de matematică și informatică competitivă. În The Fourth Conference of Mathematical Society of the Republic of Moldova: Communications in Didactics: Proceedings CMSM 4, June 28 – July 2, 2017. Chișinău: dedicated to the centenary of Vladimir Andrunachievici (1917-2017) / ed: Mitrofan Cioban, Liubomir Chiriac. – Chișinău: Tiraspol State University, 2017. 67 – 76 p. ISBN 978-9975-76-203-8
8. Corlat S., Gremalschi A. Metodologia rezolvării problemelor de geometrie computațională, Universitatea de Stat din Tiraspol, 2015.
9. Globa A. Abordări metodice privind implementarea unor tehnici de programare prin prisma complexității algoritmilor. În: Acta et Commentationes. Științe ale Educației. Revistă științifică Nr.1(4) (2014). Chișinău: Universitatea de Stat din Tiraspol, 2014. p.41-49. ISSN 1857-0623.