

## Sfaturi de bună practică

### pentru concurenții OJI / ONI 2017

Elevii care vor participa la Olimpiada Județeană/Națională de Informatică trebuie să aibă în vedere că:

- A) Codul surselor trebuie să **respecte Standardul C++**, respectiv **Standardul Free Pascal** (cu alte cuvinte, să fie corect din punct de vedere al limbajului).

### **IMPORTANT!**

**Atât la OJI 2017 cât și la ONI 2017, se admite cod sursă în acord cu Standardul C++11.**

- B) Vitezele de citire/scriere a datelor pot să difere de la o distribuție GNU C la alta.
- C) Există reguli de formatare a fișierelor de test la concursurile de algoritmică.

### A) Respectarea standardului limbajului.

Codul scris în conformitate cu standardul limbajului este un **cod portabil** și va fi compilat fără erori cu oricare compilator GNU C.

Ne vom referi în special la limbajele C/C++, deoarece în cazul acestora apar cele mai multe situații de practică defectuoasă.

1. **Nu salvați soluțiile cu extensia .c, decât dacă sunteți siguri că NU doriți să utilizați biblioteca C++.**

Fișierele cu extensia **.cpp** vor fi compilate conform standardului C++ (ISO C++ 11), iar cele cu extensia **.c** vor fi compilate conform standardului C (ISO C 99). În exemplu, sursa. c nu va compila, în timp ce sursa. cpp compilează:

sursa.c	sursa.cpp
<pre>// Se foloseste biblioteca C #include &lt;stdio.h&gt; // OK  // <b>NU</b> puteti folosi Biblioteca C++ !! <b>#include &lt;fstream.h&gt;</b> // Eroare  struct A { }; <b>A x;</b> // Eroare (standardul C)  int main() {     // variabila locala <b>i</b> in for     for (<b>int i</b> = 0; i &lt; 10; ++i) //Eroare         printf("%d", i);      return 0; }</pre>	<pre>// Se foloseste biblioteca C #include &lt;stdio.h&gt; // OK  // Se foloseste Biblioteca C++ <b>#include &lt;fstream.h&gt;</b> // OK  struct A { }; <b>A x;</b> // OK (standardul C++)  int main() {     // variabila locala <b>i</b> in for     for (<b>int i</b> = 0; i &lt; 10; ++i) // OK         printf("%d", i);      return 0; }</pre>

## 2. Header-e

### IMPORTANT!

- În limbajul C++, heder-ele cu extensia .h nu mai pot fi folosite, decât pentru header-e moștenite din limbajul C.
- Compilatoarele utilizate la OJI/ONI 2017 nu suportă stilul vechi de declarare

### Exemple:

Corect	Incorect
<pre>#include &lt;fstream&gt; // header-e C++ #include &lt;iostream&gt; #include &lt;iomanip&gt;  #include &lt;cmath&gt; // header-e C #include &lt;cstdio&gt; #include &lt;cstring&gt; // sau #include &lt;math.h&gt; #include &lt;stdio.h&gt; #include &lt;string.h&gt; <b>using namespace std;</b></pre>	<pre>// deprecated (incorect) #include &lt;fstream.h&gt; #include &lt;iostream.h&gt; #include &lt;iomanip.h&gt;</pre>

**Observație:** Pentru header-ele moștenite din C, standardul C++ acceptă (deocamdată) declarații cu extensia .h : <stdio.h>, <math.h>, <string.h>, etc.

## B). Vitezele operațiilor de intrare-ieșire în limbajele C/C++

Pentru o distribuție GNU C dată, operațiile de citire/scriere C și C++ diferă uneori în ce privește viteza de executare.

Funcțiile **scanf()**, **printf()** de pildă, sunt pentru anumite distribuții, mai rapide decât operațiile de inserție (<<) sau extracție (>>) din stream-uri, în timp ce pentru alte distribuții lucrurile stau exact invers !

**Concurenții sunt sfătuiți să studieze mediile de lucru pentru OJI/ONI și să aleagă acele metode de citire/scriere pe care le consideră optime.**

### 1. OJI 2017.

Compilatorul mediului **Code::Blocks 13.12** are particularitatea că produce executabile pentru care **vitezele de citire-scriere cu funcții C sunt mai rapide decât operațiile similare cu stream-uri.**

### 2. ONI 2017

Pentru mediile instalate la ONI 2017, (Linux Ubuntu 12.04: **gcc 4.6.x**, W7 sau Vista: **gcc 4.6**) s-au facut teste de viteză, în urma cărora se poate concluziona:

- În Linux e mai rapidă citirea și scrierea cu ajutorul stream-urilor decât cu ajutorul funcțiilor C.
- În Windows scrierea are loc cu viteze aproximativ egale, însă citirea este mai rapidă cu ajutorul funcțiilor C.

**IMPORTANT!** NU folosiți `endl` . Utilizați `'\n'` .

La scrierea în streamuri, datele de ieșire se acumulează într-un buffer (stream) care se golește periodic, nefiind nevoie de accesarea discului la fiecare operație de scriere. Manipulatorul de format `endl`, face *flush* stream-ului de ieșire, ceea ce forțează scrierea pe disc. Dacă aceasta se întâmplă într-un ciclu, atunci viteza scade catastrofal.

### C) Formatul fișierelor

1. La OJI și ONI, ca de altfel la toate competițiile de algoritmică naționale sau internaționale, este o practică curentă aceea ca **ultima linie din fișierele de test de intrare, cât și din cele de ieșire, să se termine cu caracterul newline**. Concurenții trebuie să țină seama de acest lucru.
2. **Salvați sursele doar cu numele și extensiile enumerate în enunțul problemelor.**
3. **Respectați formatul fișierului de de ieșire!**  
De exemplu, fie o problemă care are două cerințe: a) și b). Să presupunem că se cere ca răspunsul pentru cerința a) trebuie se găsească pe linia 1, iar răspunsul pentru cerința b) să se găsească pe linia 2. În situația în care nu ați reușit să rezolvați cerința a) dar aveți un răspuns pentru b), veți scrie răspunsul pentru cerința b) pe linia 2 și nu pe prima linie !
4. **FOARTE IMPORTANT!**  
La OJI ediția 2017, problemele pot avea una sau mai multe cerințe. Pentru unele probleme, comisia poate decide ca prima cerință să fie evaluată separat de celelalte, pentru ca un TLE (Time Limit Exceeded) la a doua cerință, să nu afecteze punctajul pentru prima cerință.

#### Model de enunț:

**Problema 1 - numere**

**100 puncte**

Se dau două numere naturale  $a$  și  $b$ .

#### Cerință

Să se determine :

1. Numărul de divizori ai lui  $a$ .
2. Toate numerele prime care sunt cuprinse în intervalul  $[1, b]$ .

### Date de intrare

Fișierul de intrare *numere.in* conține pe prima linie un număr natural **p**. Pentru toate testele de intrare, numărul **p** poate avea doar valoarea **1** sau valoarea **2**.

Pe linia a doua a fișierului *numere.in* se găsesc două numere naturale **a** și **b**, separate printr-un singur spațiu.

### Date de ieșire

Dacă valoarea lui **p** este **1**, se va rezolva numai punctul **1**. din cerință.

În acest caz, în fișierul de ieșire *numere.out* se va scrie un singur număr natural **n**, reprezentând numărul de divizori ai lui **a**.

Dacă valoarea lui **p** este **2**, se va rezolva numai punctul **2**. din cerință.

În acest caz, fișierul de ieșire *numere.out* va conține pe linii diferite, toate numerele prime din intervalul  $[1, b]$ . Numerele vor fi scrise câte unul pe linie, în ordine crescătoare.

### Restricții și precizări

D)  $1 \leq a, b \leq 1000$

E) Pentru rezolvarea corectă a primei cerințe se acordă **20** de puncte, iar pentru cerința a doua se acordă **80** de puncte.

F) Pentru primele **50%** din testele care verifică cerința **2**,  $a, b \leq 100$

### Exemple

numere.in	numere.out	Explicație
1 4 5	3	$p = 1, a = 4$ <b>Atenție! Pentru acest test se rezolvă doar cerința 1.</b> Divizori lui 4 sunt: 1, 2, 4  Se observă că în acest caz valoarea $b = 5$ nu se utilizează.

numere.in	numere.out	Explicație
2 3 8	2 3 5 7	$p = 2, b = 8$ <b>Atenție! Pentru acest test se rezolvă doar cerința 2.</b> Numerele prime din intervalul $[1, 8]$ sunt 2, 3, 5, și 7

**Timpi maxim de execuție: 1 secundă/test.**

**Memorie totală disponibilă 4 MB, din care 2 MB pentru stivă**

**Dimensiunea maximă a sursei: 5 KB.**