Sfaturi de bună practică

pentru concurenţii OJI / ONI 2015

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

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

IMPORTANT!

Atât la OJI 2015 cât și la ONI 2015, se admite cod sursă în acord cu Standardul C++11.

B) Vitezele de citire/scriere a datelor pot să difere de la o distribuţie GNU C la alta.

C) Există reguli de formatare a fişierelor de test la concursurile de algoritmică.

A) Respectarea standardului limbajului.

Codul scris în conformitate cu standardul limbajului este un **cod portabil** şi 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 situaţii de practică defectuoasă.

1. **Nu salvaţi soluţiile cu extensia .c, decât dacă sunteţi siguri că NU doriţi să utilizaţi biblioteca C++.**
   Fişierele cu extensia **.cpp** vor fi compilate conform standardului **C++** (ISO C++ 98), 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`

```
// Se foloseste biblioteca C
#include <stdio.h>    // OK

// NU puteti folosi Biblioteca C++ !!
#include <fstream.h> // Eroare

struct A { };
A x; // Eroare (standardul C)

int main()
{
   // variabila locala i in for
   for (int i = 0; i < 10; ++i) //Eroare
      printf("%d", i);

   return 0;
}
```

`sursa.cpp`

```
// Se foloseste biblioteca C
#include <stdio.h>     // OK

// Se foloseste Biblioteca C++
#include <fstream.h>   // OK

struct A { };
A x; // OK (standardul C++)

int main()
{
   // variabila locala i in for
   for (int i = 0; i < 10; ++i) // OK
      printf("%d", i);

   return 0;
}
```

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 2015 nu suportă stilul vechi de declarare**

Exemple:

Corect

```
#include <fstream> // header-e C++
#include <iostream>
#include <iomanip>

#include <cmath>  // header-e C
#include <cstdio>
#include <cstring>
// sau
#include <math.h>
#include <stdio.h>
#include <string.h>
using namespace std;
```

Incorect

```
// deprecated (incorect)
#include <fstream.h>
#include <iostream.h>
#include <iomanip.h>
```

Observaţie: Pentru headerele moștenite din C, standardul C++ acceptă (deocamdată) declarări cu extensia .h : `<stdio.h>`, `<math.h>`, `<string.h>`, etc.

B). Vitezele operaţiilor de intrare-ieşire în limbajele C/C++

Pentru o distribuţie GNU C dată, operaţiile de citire/scriere C şi C++ diferă uneori în ce privește viteza de executare.
Funcţiile **`scanf()`**, **`printf()`** de pildă, sunt pentru anumite distribuţii, mai rapide decât operaţiile de inserţie (<<) sau extracţie (>>) din stream-uri, în timp ce pentru alte distribuţii lucrurile stau exact invers !

Concurenţii sunt sfătuiţi să studieze mediile de lucru pentru OJI/ONI şi să aleagă acele metode de citire/scriere pe care le consideră optime.

1. OJI 2015.

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

2. ONI 2015

Pentru mediile instalate la ONI 2015, (Linux Ubuntu 12.04: **gcc 4.6.3**, W7 sau Vista: **gcc 4.6**), s-au facut teste de viteză, în urma cărora se poate concluziona :

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

IMPORTANT! **NU folosiţi `endl`. Utilizaţi `'\n'`.**

La scrierea în streamuri, datele de ieşire se acumulează într-un buffer (stream) care se goleşte periodic, nefiind nevoie de accesarea discului la fiecare operaţie de scriere. Manipulatorul de format endl, face *flush* stream-ului de ieşire, ceea ce forţează scrierea pe disc. Dacă aceasta se întamplă într-un ciclu, atunci viteza scade catastrofal:

C) Formatul fişierelor

1. La OJI şi ONI, ca de altfel la toate competiţiile de algoritmică naţionale sau internaţionale, este o practică curentă aceea ca **ultima linie din fişierele de test de intrare, cât şi din cele de ieşire, să se termine cu caracterul newline.** Concurenţii trebuie să ţină 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 2015, 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`.

Cerinţă
Să se determine :
1. Numărul de divizori ai lui `a`.
2. Toate numerele prime care sunt cuprinse în intervalul `[1, b]`.

Date de intrare
Fişierul de intrare *numere.in* conţine 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 ieşire

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

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

Restricţii şi precizări

D) **1 ≤ a, b ≤ 1000**

E) Pentru rezolvarea corectă a primei cerinţe 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 ≤ 100**

Exemple

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

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

Timp maxim de execuţie: 1 secundă/test.
Memorie totală disponibilă 4 MB, din care 2 MB pentru stivă
Dimensiunea maximă a sursei: 5 KB.