Sfaturi de bună practică

pentru concurenţii Olimpiadei Naţionale de Informatică
Constanţa 2010

Elevii care vor participa la Olimpiada Naţională de Informatică trebuie să aibă în vedere următoarele:

A) Pentru evitarea erorilor la compilare, codul surselor trebuie să **respecte Standarul C++**, respectiv S**tandardul Free Pascal** (cu alte cuvinte, să fie corect din punct de vedere al limbajului)

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

C) Faptul că 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**. Oricare compilator GNU C îl va compila.
Ne vom referi mai mult la limbajele C/C++, deoarece in cazul acestora apar cele mai multe situaţii de practică defectuoasă.

1. 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). **Nu salvaţi soluţiile cu extensia .c, decât dacă sunteţi siguri că NU doriţi să utilizaţi biblioteca C++.**
   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

Standardul limbajul C++ defineşte 33 de headere, conform tabelului:

<algorithm>	<iomanip>	<list>	<queue>	<streambuf>
<bitset>	<ios>	<locale>	<set>	<string>
<complex>	<iosfwd>	<map>	<sstream>	<typeinfo>
<deque>	<iostream>	<memory>	<stack>	<utility>
<exception>	<istream>	<new>	<stdexcept>	<valarray>
<fstream>	<iterator>	<numeric>	<sstream>	<vector>
<functional>	<limits>	<ostream>		

Facilităţile **Bibliotecii C Standard** sunt furnizate de 18 headere adiţionale:

<cassert>	<ciso646>	<csetjmp>	**<cstdio>**	<ctime>
<cctype>	<climits>	<csignal>	<cstdlib>	<cwchar>
<cerrno>	<clocale>	<cstdarg>	**<cstring>**	<cwctype>
<cfloat>	**<cmath>**	<cstddef>		

IMPORTANT!
Heder-ele cu extensia .h sunt *deprecated* pentru limbajul C++. Aceasta înseamnă că deşi unele compilatoare (cum este şi cel de la ONI 2010) mai suportă stilul vechi de declarare, compilatoarele de ultimă generaţie nu mai acceptă!

Exemple:

Corect	Deprecated (pentru headere C++)

`#include <fstream> // header-e C++`
`#include <iostream>`
`#include <iomanip>`
 	
`#include <cmath>  // header-e C`
`#include <cstdio>`
`#include <cstring>`
`using namespace std;`
 	
// directiva **using** precizeaza spaţiul de nume **std**
// în care este definită Biblioteca Standard C++
	`// deprecated`
	`#include <fstream.h>`
	`#include <iostream.h>`
	`#include <iomanip.h>`
	
	`// Corect (nondeprecated)`
	`#include <math.h>`
	`#include <stdio.h>`
	`#include <string.h>`

Observaţie: Headerele C, standardul 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 ca rapiditate.
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 ONI şi să aleagă acele metode de citire/scriere pe care le consideră optime.

Pentru mediile instalate la ONI 2010, (Ubuntu 8.04: **gcc 4.4.2**, Windows XP: **gcc 3.3.1**), am facut teste de viteză, aşa cum se poate vedea în tabelul de mai jos:

Ubuntu 8.04
gcc 4.2.4
Procesor: Intel Dual CoreE2200, @2.20GHz/ Core

Iteratii	<cstdio>		<fstream>	
	fscanf()	fprintf()	>>	<<
1000000	0,240 sec	0.170 sec	0,19 sec	0.29 sec
10000000	2.35 sec	1.85 sec	1.65 sec	3.15 sec

S-a rulat fiecare caz de zece ori şi s-a luat media.
Compilare: **gcc/g++ -lm -O2 -static -Wall numfis.cpp**

Concluzie:
E mai rapidă citirea din streamuri cu operatorul **>>** şi scrierea cu funcţia **`fprintf()`**.

Windows XP SP2
gcc 3.3.1 (pachetul OJI)
Procesor: Intel Dual CPU T2370, @1.73GHz/Core

Iteraţii	<cstdio>		<fstream>	
	fscanf()	fprintf()	>>	<<
1000000	0,51 sec	0.60 sec	3,72 sec	0.65 sec
10000000	5.47 sec	6.95 sec	35.2 sec	7.10 sec

S-a rulat fiecare caz de zece ori şi s-a luat media.
Compilare: **gcc/g++ -O2 -Wall numfis.cpp**

Concluzie: Dacă scrierea se face cu viteze aproximativ egale, în schimb, în ce priveşte citirea, **fscanf()** e mult mai rapidă decât citirea dateor din stream-ul de intrare cu operatorul >>.

ATENTIE! **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* le stream-ul de ieşire, ceea ce forţează scrierea pe disc. Daca aceasta se intamplă într-un ciclu, atunci viteza scade catastrofal:

Ubuntu 8.04,
gcc 4.2.4,
AMD Athlon(TM) XP 2500+

`#include <fstream>`
`using namespace std;`
 	
`int main()`
`{`
`  ofstream fout("numere.in");`
`  int n = 10000000;`
`  for (int i = 0; i < n; ++ i)`
`  fout << 7 << '\n';`
`  fout.close();`
`}`
	`#include <fstream>`
	`using namespace std;`
	
	`int main()`
	`{`
	`  ofstream fout("numere.in");`
	`  int n = 10000000;`
	`  for (int i = 0; i < n; ++ i)`
	`  fout << 7 << endl;`
	`  fout.close();`
	`}`

Timp de executare: **3,21** secunde	Timp de executare: **40.5** secunde !!!!

C) Formatarea fişierelor

La Olimpiada Naţională de Informatică, 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,