Anunț

Începând cu anul 2021, e posibil ca unele probleme date la InfoPro, Grupa B (urmând exemplul Grupei A), să aibă un format neobișnuit pentru mulți dintre concurenți. În acest articol vă vom prezenta noul format într-o manieră educativă.

Pe scurt, ce e nou?

Sursa trimisă nu va mai conține funcția main() și nu va mai interacționa cu fișierele de intrare și ieșire. Submisiile sunt compilate împreuna cu o anume sursă *grader.c(pp)*, oferită de comisie. Această sursă *grader.c(pp)* este ascunsă concurenților. Sursa aceasta e cea care conține funcția main(). Comisia a implementat, în *grader.c(pp*), citirea și afișarea. Cu toate acestea, sursa *grader.c(pp)* nu va compila și nu va rezolva problema de una singură. Ea apelează una sau mai multe funcții care trebuie implementate de către concurent.

Să notăm sursa trimisă cu *submit.c(pp)*, deși orice denumire cu extensia *c(pp)* e un nume la fel de valid. Sursa *submit.c(pp)* va conține aceste funcții necesare și implementarea lor. Sursa *submit.c(pp)* este o sursa **C(++)** obișnuită, în care se pot implementa funcții noi, **se pot folosi variabile globale** și se pot include biblioteci și defini **macro**uri. Marile excepții sunt că declararea funcției *main()* va rezulta într-o eroare de compilare, respectiv că citirea și afișarea cu fișiere poate rezulta într-un comportament greșit al programului.

Un exemplu

Antrenamentul Grupei B conține problema **Sum**, care este un exemplu de problemă cu noul format. Vă recomandăm să experimentați diferite submisii aici, înainte de următoarea rundă.

O posibilă sursă *submit.c(pp)* care obține 100 de puncte conține o singură linie:

```
int sum(int a, int b) { return a + b; }
```

Așa cum enunțul specifică în secțiunea „**Protocol de interacțiune**”, concurentul trebuie să **nu** implementeze funcția main(). În rest, e liber să implementeze cum dorește funcția sum(). Pentru a exemplifica aceasta, am mai pregătit o sursă *submit.cpp*, de data aceasta formată din 17 linii, care obține 100 de puncte:

```
#include <cstdio>
#include <vector>
#define ELEMENT_NEUTRU 0
int adun(std::vector < int >& v) {
  int suma = ELEMENT_NEUTRU;
  for (auto& a : v) {
    suma += a;
    printf("%d\n", suma); /// debugging...
  }
  return suma;
}
```

```
int sum(int x, int y) {
   std::vector < int > v;
   v.push_back(x);
   v.push_back(y);
   return adun(v);
}
```

Modelul sursei *grader.c(pp)*

Toate problemele cu noul format vor conține, între atașamente (în timpul concursului, le găsiți la secțiunea „Statement"), două fișiere: **grader.c** și **grader.cpp**. Cele două fișiere nu sunt decât niște atașamente, nu sunt neapărat identice cu sursele *grader.c* și *grader.cpp*, folosite la evaluarea surselor trimise. Dar, din multe puncte de vedere, mai ales al funcționalității și al relației cu sursele concurenților, **grader.c(pp)** este respectiv echivalent cu *grader.c(pp)*.

În particular, sursa **grader.c(pp)** conține semnăturile funcțiilor care trebuiesc implementate în *submit.c(pp)*. Aceste semnături mai apar atât în enunțul problemei, cât și în *grader.c(pp)*. Concurenții pot modifica, pe propriile calculatoare, fișierul grader.c(pp) după cum doresc. De asemenea, pot descărca atașamentele de pe interfață de mai multe ori.

Modelul atașat grader.c(pp) poate fi diferit de sursa ascunsă grader.c(pp) în mai multe aspecte. Iată două exemple:

- Sursa ascunsă poate parsa fișierul de intrare, pentru a face evaluarea mai rapidă. Dar modelul atașat va citi datele de intrare în mod standard, pentru a face sursa mai ușor de citit;
- Sursa ascunsă poate conține cod relevant pentru rezolvarea problemei. Modelul atașat, în schimb, nu va implementa nimic care să-i ajute pe concurenți să găsească soluția problemei.

Garantăm că dacă *submit.c(pp)* compilează împreună cu **grader.c(pp)**, va compila și împreună cu *grader.c(pp)*.

Testarea locală a sursei trimise

Pentru experimente locale și pentru a testa *submit.c(pp)* pe calculatorul propriu, e suficient ca sursele **grader.c(pp)** și *submit.c(pp)* să fie compilate împreună. Astfel, concurentul poate experimenta cu programul obținut ca în problemele cu format obișnuit. Urmează să prezentăm trei metode recomandate pentru a compila și rula cele două surse.

1. Code::Blocks and other Integrated Development Environments

   În formatul obișnuit, putem compila și rula o sursă care conține funcția main() dintr-un proiect care conține această sursă, alături de fișierele de intrare și ieșire. Pentru noul format, nu e nevoie decât ca, în loc să fie o singură sursă asociată proiectului, să fie amândouă prezente în câmpul „**Sources**".

2. Terminal. Ubuntu & Co

   În formatul obișnuit, putem compila o sursă prin comanda `gcc main.c` sau `g++ main.cpp`. Apoi rulăm prin comanda `./a.out`. Pentru noul format, nu e nevoie decât să compilăm ambele surse deodată: `gcc grader.c submit.c` sau `g++ grader.cpp submit.cpp`. Pentru a rula programul, comanda rămâne neschimbată: `./a.out`.

3. E suficientă o sursă

```
#define SUBMIT
#ifndef SUBMIT
#include<fstream>
using namespace std;
static int a, b;
int sum(int a, int b);
ifstream fin("sum.in");
ofstream fout("sum.out");
int main(){
    fin>> a >> b;
    fout<< sum(a, b);
    return 0;
}
#endif // SUBMIT
#include <algorithm>
int sum(int n, int m) {
    int mn = std::min(n, m);
    int mx = std::max(n, m);
    return mn + mx;
}

```

O alternativă este folosirea unei singure surse. Ea începe cu #define SUBMIT, apoi încadrează conținutul sursei **grader.c(pp)** între liniile #ifndef SUBMIT și #endif, iar la final are codul sursei *submit.c(pp)*. Aceasta poate juca direct rolul lui *submit.c(pp)*, întrucât compilatorul va ignora codul încadrat de liniile #ifndef SUBMIT și #endif. Dar atunci când prima linie, #define SUBMIT, este comentată, codul nu va mai fi ignorat, așa că sursa joaca rolul programului complet, cu care concurentul poate experimenta ca în cazul formatului obișnuit.

În imaginea din stânga, puteți vedea implementarea unei noi soluții la problema **Sum**, care exemplifică în **C++** paragraful de mai sus.

Motivație

Am vrea să subliniem câteva motive pentru care lansăm noul format.

- Mai mult control: Felul în care datele de intrare (ieșire) sunt extrase (introduse) nu vor mai putea crea diferențe între performanțele surselor trimise. Doar felul în care acestea sunt procesate în funcțiile care rezolvă cu adevărat problema va conta.
- Mai convenabil: Concurenții se vor putea concentra numai pe rezolvarea efectivă a problemei și mai puțin pe interpretarea datelor de intrare și ieșire.
- Mai aproape de standardele internaționale: Olimpiada Internațională, precum și majoritatea concursurilor si olimpiadelor regionale, au renunțat la formatul „obișnuit” în favoarea acestui nou standard, care oferă stabilitate, simplitate și eficiență.

„Frequently Asked Questions”

Pe măsură ce vom primi întrebări legate de noul format, vom actualiza anunțul cu cele mai populare întrebări la care găsim un răspuns. Primele două sunt anticipate de la prima publicare a acestui articol.

1. Dacă *grader.c(pp)* are un comportament randomizat, e posibil ca același program să se comporte diferit de la o submisie la alta? Răspuns: Garantăm că grader.c(pp) nu va avea un comportament randomizat în așa fel încât să interacționeze diferit cu submit.c(pp). Alternativa este Pseudo-Random-ul, care simulează caracterul aleator al fratelui său Random Pur, dar conservă comportamentul de la o rulare la alta a programului.
2. Ce se întâmplă atunci când numele unor funcții și ale unor variabile globale din *grader.c(pp)* coincid cu cele din *submit.c(pp)*? Răspuns: Câtă vreme *submit.c(pp)* compilează împreună cu **grader.c(pp)**, garantăm că sursa *grader.c(pp)* este scrisă în așa fel încât orice coincidență a numelor să fie tratată ca nume diferite de către compilator.