Osztott rendszerek specifikációja és implementációja 1. beadandó - Dokumentáció

Név: Soós Bálint
Neptun kód: HDX9MU
Elérhetőség: soosbalint95@gmail.com

Gyakorlatvezető: Horpácsi Dániel
Csoport: 5.
2016. november 6.

Valamikor, a nem túl távoli jövőben az Informatikai Kar túlnőtt a programozóképzés keretein. A hatalmas túljelentkezés és munkaerőpiaci igény miatt szükségessé vált további hallgatók felvétele, akik később a megszerzett tudást versenykörülmények között tudják hasznosítani. Világossá vált, hogy a profi fejlesztők és a hivatásos felhasználók együttműködése elengedhetetlen. Bár továbbra is hatalmas érdeklődés övezte a programozói képzést, ám egyre többen jöttek gamer ambíciókkal. Az egyetem úgy döntött, hogy új Kart indít a játékos közösség igényeinek kiszolgálására, hogy a profi eSportolók közössége minőségi oktatást kaphasson az ELTE-n belül.

A különböző tanszékekhez, az oktatói és hallgatói létszámhoz, valamint a géptermek felszereltségéhez azonban szükséges információ a megfelelő játékostáborral arányos erőforrások biztosítása. A proginf.-es képzésben résztvevő, B szakirányos diákokra bízzák a kiszámítását annak, hogy milyen súlyú a különböző játékok iránti érdeklődés, hogy ehhez mérve igazíthassák a tantárgyi felosztást.

A felvételizők között ezért előzetes felmérést végeztek, ki milyen játékokkal játszik szabadidejében. Az így kapott adathalmaz közkedvelt játékai közül M kapott döntő mennyiségű szavazatot, így a második körben ezekre szűkült a kérdőív, melyben minden hallgató válaszolt, hogy egy héten mennyi időt (percet) tölt az egyes játékokkal. Természetesen voltak, akik nem csupán egyetlen műfajnak hódoltak, így előfordult, hogy több kitöltés is érkezett ugyan attól a hallgatótól. A névtelenség, de egyszeri válaszadás fenntartásának érdekében a kitöltőket NEPTUN-kóddal azonosították. A játékokat egy-egy szóközt nem tartalmazó, szöveg típusú adat reprezentálja (pl. "WoW" vagy "LoL").

Bemenet

A bemenet – input.txt – első sorában tartalmazza a szóközzel elválasztott N és M pozitív egészeket – ahol következő N sorában pedig egy-egy neptunkód, játéknév és percben eltöltött idő (egész szám) szóközzel elválasztott hármasát:

```
N M
NEPTUN_1 gameazon perc   - az 1. válasz adatai
NEPTUN_2 gameazon perc   - az 2. válasz adatai
...
...
...
NEPTUN_N gameazon perc   - az N. válasz adatai
```

Feladat

A főfolyamat olvassa be az adatokat, indítson M gyerekfolyamatot, majd minden gyerekfolyamathoz társítson egy-egy játékhoz tartozó adathalmazt. (A játékok száma pontosan M, így minden játék adatát párhuzamosan kell kiszámolni.) A gyerekfolyamatok dolga megállapítani, hogy az adott játékkal mennyi időt töltöttek összesen (percben) a válaszadók, valamint az átlagos játékidő kiszámítása az adott játékhoz (egész percre lefelé kerekítve - floor(..)). Ezt a két adatot küldje vissza a szülőfolyamatnak.

A főfolyamat ezek után a játékokhoz tartozó, összesen és átlagosan eltöltött időt szóközzel elválasztva a JÁTÉKOK azonosítója ALAPJÁN BETŰRENDBEN írja az output.txt kimeneti fájlba.

Felhasználói dokumentáció

1. ***Környezet***

   A program fordítástól függően `.out` kiterjesztés esetén Linux/OSX oprendszereken, `.exe` kiterjesztés esetén Windows oprendszereken használható. Telepítésre nincs szükség, elegendő a futtatható állományt elhelyezni a számítógépen.

2. ***Használat***

   A program elindításához nincs szükség parancssori paraméterekre, így parancssoron kívül is lehet futtatni. A programmal egy mappaszinten kell elhelyezni az `input.txt` bemeneti fájlt. A program eredményét ugyanezen a szinten az `output.txt` kimeneti fájlba írja.

   Egy lehetséges bemenetet tartalmaz a mellékelt `input.txt` fájl, illetve a `tests` mappában további példák találhatóak. Saját bemeneti fájlok esetén fontos, hogy a feladatban megadott szempontok alapján írjuk az adatokat a fájlba, mivel ezek helyességét a programban nem ellenőrizzük.

Fejlesztői dokumentáció

1. ***Megoldás módja***

   A programot logikailag két részre, fő- és alfolyamatokra bonthatjuk. A főfolyamat végzi a bemeneti adatok beolvasását, amiket egy `Map` adatszerkezetben tárol. Létrehoz a `Map` hosszúságának megfelelő számú alfolyamatot, amelyek kiszámolják az adott játékkal töltött idő összegét és átlagát. Az alfolyamatokból érkező eredményeket a főfolyamat írja a kimeneti fájlba.

2. ***Implementáció***

   A beolvasott adatokat egy `Map` adatszerkezetben tároltam, ahol a kulcs a játék azonosítója, az érték egy vektor, amelyben a játékkal eltöltött időket tároltam. Egy alfolyamat a map egy vektorát kapják paraméterül, visszatérési értékük az értékek átlagából és összegéből alkotott pár.

   Az alfolyamatok eredményét szintén egy map-ban tároltam. A C++11 szabvány

nyelvi elemeit kihasználva a map értékei `std::future<std::pair<int, int»` típusúak. A szükséges M számú (játékok száma, amely megfelel a map méretével) alfolyamatot az `std::async` segítségével párhuzamosítottam. A teljes implementáció egyetlen forrásfájlba szervezve, a `main.cpp` fájlban található.

3. ***Fordítás***

   A program forráskódja a `main.cpp` fájlban található. Fordításához egy C++11 szabványt támogató fordítóprogram szükséges. A fejlesztéshez a `g++` fordítót használtam: `g++ -std=c++11 main.cpp`

4. ***Tesztelés***

   Teszteléshez a `tests` mappában található bemeneti fájlokat használtam, amelyek mindegyikére az elvárt kimenetet állítja elő a program.