Orsi beadandó feladat - dokumentáció

2016.11.05.

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").

1 Feladat

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

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.

//egy példa sor az output fájlból:
//WoW 6000 573

További elvárások:

A megoldásaitokat egyetlen ZIP fájlba tömörítve töltsétek fel! Kérünk benneteket, hogy csak a szükséges forrásfájl(oka)t rakjátok bele, ne teljes projektet (.exe semmiképp sem)!

A különböző technológiák miatt szeretnénk mindenkit megkérni, hogy az alábbiak vegye figyelembe a feltöltés során:
PVM: A két .cpp fájlt, és a Makefile.aimk-t egyaránt csomagoljátok be, a szülőfolyamathoz tartozó kód 'master.cpp' -nek legyen elnevezve.
(C++11: Nincs külön megkötés, tetszőleges fájlnévvel rendelkezhet.)

2 Felhasználói dokumentáció

2.1 Környezet

A program több platformon futtatható, nincsen dinamikus függősége. Telepítésre nincs szükség, elegendő a futtatható állomány elhelyezni a számítógépen.

2.2 Használat

A program elindítása egyszerű, mivel nem vár parancssori paramétereket, így parancssoron kívül is lehet futtatni. A fájl mellett kell elhelyezni az input.txt-t 1 fájl, melyet feldolgoz és az eredményt az "output.txt" nevű fájlba írja, a bemeneti sorrend alapján. Egy lehetséges bemenetet tartalmaz a mellékelt input.txt tesztfájl. Saját bemeneti fájlok esetén fontos, hogy a feladatban megadott szempontok alapján írjuk az adatokat az inputfájlba, mivel a program kilép ha az adatok a fájlban nem helyesek.

3 Fejlesztői dokumentáció

3.1 A megoldás módja

A kódot logikailag két részre bonthatjuk, egy fő és több gyermekfolyamatra. A főfolyamatot a master.cpp-ben szereplő main függvény fogja megvalósítani. Feladata, hogy beolvassa az inputfájl tartalmát, majd a feladatban meghatározott módon létrehozzon alfolyamatokat.

Az N sor szavazat, illetve a M fajta csoport, meghatározó két pozitív egész szám, beolvasása után. Egy olyan struktúrába olvassuk be az adatokat, amely támogatja, az adatok csoportosítását kulcs szerint. Erre szükség lesz, amikor kiosztjuk a gyerekfolyamatoknak, a részfeladatokat. Egy gyermek részfeladata, hogy meghatározza a hozzá tartozó kategória, összes illetve átlagos idejét, majd visszaüzenje a szülőnek.

A főfolyamat az alfolyamatokból visszanyert adatot, rendezett formában az output.txt fájlba írja bele.

3.2 Implementáció

A C++ megvalósítás során, a nyelvi elemek közül, a legmegfelelőbb adatstruktúra az **std::map,** azon belül is a **std::multimap**, mivel a map egy asszociatív adatstruktúra, amely tartja a rendezettségét a kulcsai szerint növekvő sorrendben. (Játéknévsorrend) Továbbá egyedi kulcsokkal kell ellátni az egyes elemeket. Ezt kiküszöbölendően a multimap már megenged több elemet ugyanahhoz a kulcshoz, és ugyanúgy tartja a rendezettséget.
Az implementáció a master.cpp szülőfolyamatból és a child.cpp gyerekfolyamatok main függvényeinek törzsében található.

3.3 Fordítás

A program forráskódja a master.cpp illetve a child.cpp. A program fordításához követelmény egy c++11 szabványt támogató fordítóprogram megléte a rendszeren.

A fejlesztés, ill. tesztelés során a g++ fordítót használtam. A fordítóban speciális beállításként a stringeket támogató -Wno-write-strings kapcsolót is alkalmazni kell.

Az –std=c++11 kapcsolóm is szükséges, mert alapértelmezetten régebbi c++ szabványt támogat a fordító.

A programhoz csatolt Makefile.aimk fájl segítséget nyújt a fordításhoz, mert magába foglalja a fordításhoz szükséges információkat.

3.4 Tesztelés

A program helyes működését, különböző a feladathoz kapott tesztelési anyagok segítségével végeztem. A minták általános, és pár szélsőséges eseteket dolgoznak fel.

1. Input_1.txt

11 7
D482J4 LoL 606
D482J4 OW 857
BZJW9I HotS 2551
LFQSJN DotA 2137
LFQSJN LoL 1667
U4NOA1 CoD 1354
U4NOA1 MC 1277
TYGS6Y CoD 1442
AEJRLZ GTAV 182
ED06UQ DotA 1264
6VSJ3K MC 193

2. Output_1.txt

CoD 2796 1398
DotA 3401 1700
GTAV 182 182
HotS 2551 2551
LoL 2273 1136
MC 1470 735
OW 857 857

3. Input_2.txt

6 5
9MQGI1 MC 61
9MQGI1 WoW 1564
NC2XBQ LoL 2378
NC2XBQ GTAV 733
EMV17T WoW 387
JXJJTU HotS 352

4. Output_2.txt

GTAV 733 733
HotS 352 352
LoL 2378 2378
MC 61 61
WoW 1951 975

5. Input_3.txt

2 2
PXXMYG OW 1319
PXXMYG CoD 2440

6. Output_3.txt

CoD 2440 2440
OW 1319 1319

7. Input_4.txt

1 1
PXXMYG WoW 1319

8. Output_4.txt

WoW 1319 1319