Programozás II. 1. ZH

SZTE Szoftverfejlesztés Tanszék

2022. ősz

Technikai ismertető

- A programot C++ nyelven kell megírni.
- A megoldást a *Bíró* fogja kiértékelni.
  - A Feladat beadása felületen a Feltöltés gomb megnyomása után ki kell várni, amíg lefut a kiértékelés. **Kiértékelés közben nem szabad az oldalt frissíteni vagy a Feltöltés gombot újból megnyomni** különben feltöltési lehetőség veszik el!
- Feltöltés után a *Bíró* a programot `g++` fordítóval és a
  `-std=c++17 -static -O2 -DTEST_BIRO=1`
  paraméterezéssel fordítja és különböző tesztesetekre futtatja.
- A program működése akkor helyes, ha a tesztesetek futása nem tart tovább 5 másodpercnél és hiba nélkül (0 hibakóddal) fejeződik be, valamint a program működése a feladatkiírásnak megfelelő.
- A *Bíró* által a `riport.txt`-ben visszaadott lehetséges hibakódok:
  - Futási hiba 6: Memória- vagy időkorlát túllépés.
  - Futási hiba 8: Lebegőpontos hiba, például nullával való osztás.
  - Futási hiba 11: Memória-hozzáférési probléma, pl. tömb-túlindexelés, null pointer használat.
- A riport.txt és a fordítási log fájlok megtekinthetőek az alábbi módon:
- A programot 20 alkalommal lehet benyújtani, a megadott határidőig.
- A programban szerepelhet `main` függvény, amely a pontszámításkor nem lesz figyelembe véve. Azonban ha fordítási hibát okozó kód van benne az egész feladatsor 0 pontos lesz.

Általános követelmények, tudnivalók

- Csak a leírásban szereplő osztályokat, metódusokat és adattagokat kell megvalósítani, egyéb dolgokért nem jár plusz pont.
- Minden metódus, amelyik nem változtatja meg az objektumot, legyen konstans! Ha a paramétert nem változtatja a metódus, akkor a paraméter legyen konstans!
- string összehasonlításoknál az egyezés a pontos egyezést jelenti, azaz ha kis-nagy betűben térnek el, akkor már nem tekinthetők egyenlőnek (pl. a "piros" != "Piros")

- A leírásokban bemutat példákban a string-ek köré rakott idézőjelek nem részei az elvárt kimenetnek, azok csak a string határait jelölik. Például ha az szerepel, hogy a példa bemenetre az elvárt kimenet az, hogy `"3 alma"`, akkor az elvárt kimenet idézőjelek nélkül az `3 alma`, de a szóköz szükséges!
  - A tesztesetekben nem lesz ékezetes szöveg kiíratása.
- Az elvárt kimeneteknek karakterről karakterre olyan formátumúnak kell lennie, ami a feladatban le van írva (szóközöket és sortöréseket is beleértve).

Kezdés

A feladat elkezdéséhez a minta.zip-ben található forráskódot kell megnyitni, és azt kell módosítani, kiegészíteni.

1. feladat: Átmegyek prog2-ből! (5 pont)

Készítsd el a **TulKevesPont** kivétel osztályt, ami publikusan öröklődik az std::exception osztályból. **(1+1 pont)**

A konstruktor egy lebegőpontos számot vár paraméterben, ami alapján beállítja a hibaüzenetet.

- Ha 50-nél kisebb a paraméter értéke, akkor "Megbuktam! :(" legyen a hibaüzenet,
- különben, ha 65-nél kisebb, akkor "Meglett a kettes!",
- minden egyéb esetben pedig "A kettesnel nincs jobb, csak szebb" a beállítandó szöveg. **(0+1 pont)**

Definiáld felül a *what* metódust, amely visszaadja a megalkotott üzenetet. **(0+2 pont)**

2. feladat: Éljük túl! (7 pont)

Valósítsd meg a mintában található *jegyek_szama* függvényt!

A függvény paraméterben egy map-et kap, ami az egyes diákok eredményét tartalmazza. A map kulcsa a diák neve (egyedi), az értéke pedig a prog2 jegye.

A függvény első körben határozza meg a diákok jegyeinek átlagát. Amennyiben az átlag 85 alatt van, akkor dobjunk egy **TulKevesPont** kivételt, amit a jegyek átlagával inicializál.

Ezután a **count_if** függvénnyel határozzuk meg, hogy hány olyan diák van, akinek a pontszáma 85 alatti. A függvény visszatérési értéke ez az érték legyen.

Megjegyzés: Az eredményre csak akkor kapsz pontot, ha a count_if függvényt pontosan egyszer használod és helyesen hívod meg.

3. feladat: Házi (7 pont)

A `Hazi` struktúra tartalmazza, hogy egy adott félévben hány házi van, illetve azt, hogy mennyit teljesítettünk közülük. **A Hazi struktúrán ne módosíts!**

Valósítsd meg a mintában található *teljesitetteket_kitorol* függvényt. A függvény első paraméterben egy vectort kap referenciaként, amely *Hazi* objektumokat tárol. Második paramétere egy valós szám (teljesítési arány).

A függvény távolítsa el a vektorból azokat a házikat az **erase** és **remove_if** függvények segítségével, amelyek teljesítettségi aránya (teljesített házik / összes házi) nagyobb a második paraméterben érkező értéknél.

hint: ügyelj a fejlesztői környezet által adott warningokra!

Megjegyzés: Az eredményre csak akkor kapsz pontot, ha a remove_if függvényt pontosan egyszer használod és helyesen hívod meg.

4. feladat: Könyv (7 pont)

A `Konyv` struktúra tartalmazza, hogy a könyv milyen hosszú, illetve a könyv nyelvét.

Egészítsd ki a mintában található `Konyv` struktúrát a $<$ operátorral, amely az összehasonlítást a *hossz* adattag alapján végzi.

Valósítsd meg a mintában található *olvasas* függvényt. A függvény első paramétere egy könyvekből álló tömb, míg második paramétere a tömb hossza (hány könyv van a tömbben).

Készíts egy *set*-et, amelybe tedd bele a tömbben található könyveket.

Ezután a **find_if** algoritmus segítségével határozd meg, hogy van-e olyan könyv, aminek a nyelve a *magyar*. Ha igen, akkor írd ki a standard outputra a *"Konyv elolvasasa ezen a nyelven: magyar"* szöveget. Ha nincs ilyen könyv, akkor a *"Konyv elolvasasa ezen a nyelven: angol"* íródjon ki. Mind a két esetben a kiíratás után legyen sortörés is.

Megjegyzés: A feladatra csak akkor kapsz pontot, ha a find_if függvényt pontosan egyszer használod és helyesen hívod meg.

5. feladat: Kedvenc italok (7 pont)

A `KedvencItal` struktúra tartalmaz egy ember kedvenc italairól információt, mégpedig hogy kinek mi a kedvenc itala.

Valósítsd meg a mintában található *kedvenc_italok* függvényt!

A függvény egy `KedvencItal` objektumokból álló tömböt vár paraméterben, illetve a tömb hosszát.

A függvény feladata, hogy visszaadjon egy olyan vektort, ami tartalmazza az italok neveit az adott italt szerető emberek darabszáma szerinti csökkenő sorrendben. Tehát a vektor elején legyen annak az italnak a neve, ami a legtöbb embernek a kedvence, míg a vektor végén legyen annak az italnak a neve, ami a legkevesebb embernek a kedvence.

A feladat megoldása során feltehetjük, hogy minden italt más mennyiségű ember szeret.

Megjegyzés: a függvény megvalósítása során bármit használhatsz, azonban az órán tanult adatszerkezetekkel és algoritmusokkal jóval egyszerűbb a megoldás. A megoldás során további függvényeket / osztályokat... létrehozhatsz, amennyiben ez szükséges.

Hint: Több adatszerkezetet is használhatsz a függvényben.