Programozás II. ZH

SZTE Szoftverfejlesztés Tanszék

2025. 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 1. és 2. ZH során STL tárolók használata nem megengedett! Tárolók használata 0 pontot eredményez.
- Az elvárt kimeneteknek karakterről karakterre olyan formátumúnak kell lennie, ami a feladatban le van írva (szóközöket és sortöréseket is beleértve).
- Ha az objektum másolása nem triviális (azaz a fordító által generált másolás nem elegendő), akkor a megfelelő másolást is meg kell valósítani.

Zh alatt használható segédanyag

- **A ZH során használható segédanyag elérhető bíróban.**
  - https://biro.inf.u-szeged.hu/kozos/prog2/

Kiindulási projekt, megoldás feltöltése

- **A megoldáshoz az előre kiadott osztályok módosítása szükséges lehet.**
  - Nem minden ZH esetében van kiindulási projekt.
- Feltöltéskor ezeket az osztályokat is fel kell tölteni és a módosításokat is pontozhatja a bíró!
- Egyes tesztesetekben a bíró módosított osztályt is használhat ezen kiinduló osztályok helyett, ezzel tesztelve a valóban helyes működést!

1. Feladat: Járatok halmazba pakolása (4 + 8 pont)

Adott a kiindulási fájlban egy Jarat nevű osztály, ami repülőjáratokat reprezentál két város között. Egy előjeltelen egész számban tárolja a járat árát (`ar`), a `honnan` és `hova` adattagok megadják a kiindulási illetve a célvárosokat. Bővítsd úgy az osztályt, hogy be tudjuk pakolni egy halmazba. A rendezés az alábbi módon történjen:

- Ha két járat `honnan` adattagja eltér, akkor ez alapján rendezzük (lexikografikusan)
- Ha két járat ugyanonnan indul, akkor a `hova` adattag alapján rendezzünk (lexikografikusan)
- Az árat nem vesszük be a rendezési szempontok közé

Készítsd el a `jarat_set` nevű függvényt, ami a paraméterében egy járatokat tároló vektort kap és átmásolja a vektor elemeit egy halmazba. Ezzel a halmazzal térjen vissza a függvény. Természetesen előfordulhat, hogy ütközés miatt néhány járat nem kerül bele a set-be.

2. Feladat: Legolcsobb jarat (9 pont)

Készíts egy `legolcsobb` függvényt, ami paraméterében egy Járatokat tároló vektort és egy map-et vár. A map kulcsa string, az érték pedig előjeltelen egész. A függvény feladata, hogy kigyűjtse a legolcsóbb járatot két város között. A kulcsot a Járat objektumok `honnan` és `hova` adattagjaiból tudjátok előállítani összefűzéssel (konkatenáljátok össze egymás után, elválasztójel nélkül a két várost). A map-be várospáronként helyezzétek el a legolcsóbb járatok árát. Azaz a map bemenő-kimenő paraméter.
Pl. adott ez a két járat:
honnan: London, hova: Budapest, ar: 50000
honnan: London, hova: Budapest, ar: 10000
A map tartalma:
LondonBudapest: 10000

3. Feladat: Számol (9 pont)

Készíts egy **szamol** függvényt, ami egy járatokat tároló vektort és egy sztringet vár és a `count_if` segítségével megszámolja, hogy hány járat megy a sztring paraméterben kapott városba. Ez legyen a függvény visszatérési értéke. Csakis a `count_if` használata esetén jár pont a feladatra.

4. Feladat: Filter (9 pont)

Készíts egy függvényt `filter` néven, ami egy járatokat tároló vektort és két sztringet kap paraméterül (`param1` és `param2`). Visszatérési értéke egy járatokat tároló vektor. Az eredménybe először azokat a járatokat tedd bele, amik `param1`-ből `param2`-be mennek (odautak). Ezek után tedd bele a visszautakat is, vagyis azokat a járatokat, amik `param2`-ből `param1`-be mennek.

5. Feladat: Teljes rendezés (9 pont)

Készíts egy függvényt `fullSort` néven, ami egy járatokat tároló vektort kap paraméterül és a `sort` függvényt használva rendezi a járatokat ár szerint növekvő sorrendbe. A paraméterben kapott vektor kimenő-bemenő paraméter, azaz ebben végezd el a rendezést. A rendezés ne az 1-es feladatban megadott feltétel szerint, hanem kizárólag az ár alapján történjen. Az 1-es feladatban írt operátort ne módosítsd, hiszen akkor arra nem fogsz pontot kapni. Tipp: definiálj új rendezést a `sort` harmadik paraméterében.

6. Feladat:Járatok összköltsége (7 pont)

Készíts egy `fullCost` nevű függvényt, ami egy járatokat tartalmazó vektort és egy `unsigned` értéket kap paraméterként, és visszaadja a vektorban szereplő összes olyan járat árának összegét, aminek az ára nem éri el a limitet. A függvény visszatérési típusa legyen előjeltelen egész. Csakis az `accumulate` használata esetén jár pont a feladatra.

7. Feladat:Akció (9 pont)

Készíts egy `akcio` nevű függvényt, ami egy járatokat tartalmazó vektort és egy `unsigned` értéket (százalék) kap paraméterként. Feladata, hogy visszaadjon egy olyan vektort, amiben a paraméterben kapott vektor járatai a százalékban megadott értékkel le vannak akciózva. Példa: ha X járat ára 120000 és a paraméterben érkező százalék 20, akkor a leakciózott ára 120000*0.8 = 96000. Ha a paraméterben érkező százalék nagyobb volna, mint 100, legyen std::exception kivétel dobva. Csakis a `transform` használata esetén jár pont a feladatra.