HÁZI FELADAT

Specifikáció

Feladat:

A https://infocpp.iit.bme.hu/hf_otlet oldalon található feladatok közül a **Vonatjegy** nevűt választottam. A feladatleírás másolata:

Vonatjegy
Tervezze meg egy vonatjegy eladó rendszer egyszerűsített objektummodelljét, majd valósítsa azt meg! A vonatjegy a feladatban mindig jegyet és helyjegyet jelent együtt. Így egy jegyen minimum a következőket kell feltüntetni:

- vonatszám, kocsiszám, hely
- indulási állomás, indulási idő
- érkezési állomás, érkezési idő

A rendszerrel minimum a következő műveleteket kívánjuk elvégezni:

- vonatok felvétele
- jegy kiadása

A rendszer később lehet bővebb funkcionalitású (pl. késések kezelése, vonat törlése, menetrend, stb.), ezért nagyon fontos, hogy jól határozza meg az objektumokat és azok felelősségét. Valósítsa meg a jeggyel végezhető összes értelmes műveletet operátor átdefiniálással (overload), de nem kell ragaszkodni az összes operátor átdefiniálásához! A megoldáshoz **ne** használjon STL tárolót!

Specifikáció:

A vonatjegy eladó rendszer legfontosabb alapeleme maga a vonatjegy. Egy jegyen a leírásban foglaltakon kívül a következőket fogom feltüntetni:

- jegy ára, kedvezmények (ha van)
- kocsiosztály
- retúr jegy-e
- melyik állomáson adták el

A rendszer képes továbbá a feladatleírásban szereplő műveleteken felül a felvett vonatokból menetrendet generálni, vonatokat törölni és az eddigi jegyeladásokat összesíteni.

A feladat ezenfelül előírja, hogy operátorokkal valósítsak meg jegyeken értelmezhető műveleteket. Erre a következőket tervezem implementálni:

- operator== és operator!= : két jegy minden attribútumának egyenlőségét ellenőrzik
- operator> és operator< : két jegy árát hasonlítja össze
- operator<< és operator>> : jegy adatainak kiírásához vagy betöltéséhez
- operator[] : a jegy valamilyen tulajdonságának lekérdezéséhez
  (pl.: std::cout << jegy[“ár”] → 200 ft)

A laboron egyeztetettek alapján egy osztály sablon dinamikus tömbben fogom majd tárolni többek között a vonatokat és az eladott jegyek adatait.

Teszteléshez a leírt funkciókat átfogóan kihasználó tesztprogramot fogok írni. Hibakezelést az értelemszerűen elvárható helyeken fogok alkalmazni és ezt a tesztprogram is ellenőrizni fogja.

Program használata:

A program konzolos felületen használható, az elérhető funkciók számozott menüben jelennek meg és a megfelelő szám begépelésével érhetőek el.

Amennyiben egy platformon rendelkezésre áll C++ fordító, a programnak működőképesnek kell lennie.

HÁZI FELADAT

Terv

Objektumok:

A **Vonatjegy** feladat alapeleme maga a vonatjegy. Ezt a *Jegy* osztályban fogom megvalósítani. Továbbá a *Vonat,* a gyakorlatokon elkészített *String* és megegyezés alapján a *dyn_array<T>* osztályokból fog állni a programom.

Jegy osztály:

A *Jegy* osztály mezői és metódusai a feladatleírásnak megfelelően lettek létrehozva. A vonatszám, indulási állomás, indulási idő, érkezési állomás és érkezési idő egy *Vonat* objektumra mutató pointeren keresztül érhető el mivel ezeket az információkat a *Vonat* osztály tartalmazza. A Kedvezményeknek a *Jegy* osztály csupán a megnevezését tartalmazza, ezek megadása és figyelembe vétele a jegy kiadása során történik. A mezők elérésére az operator[] használható ez azonban írásvédett, mezők módosításához a get<T>(const String&) metódus használható az igényelt attribútum nevének megadásával.

A csatolt osztálydigramon konstruktorok nem lettek feltüntetve. Az osztály rendelkezik másoló konstruktorral és egy olyan konstruktorral amely minden attribútumának (felülírhatóan) ad értelemszerű alapértéket.

Vonat osztály:

A *Vonat* osztály tartalmazza a következő információkat: vonatszám, indulási állomás, indulási idő, érkezési állomás és érkezési idő. A mezők elérése a *Jegy* osztályban ismertetett módon történik.

A csatolt osztálydigramon konstruktorok nem lettek feltüntetve. Az osztály rendelkezik másoló konstruktorral és egy olyan konstruktorral amely minden attribútumának (felülírhatóan) ad értelemszerű alapértéket.

String osztály:
A *String* osztály az ötödik laborfeladatban elkészítettel teljes mértékben megegyezik. *Részletek itt*

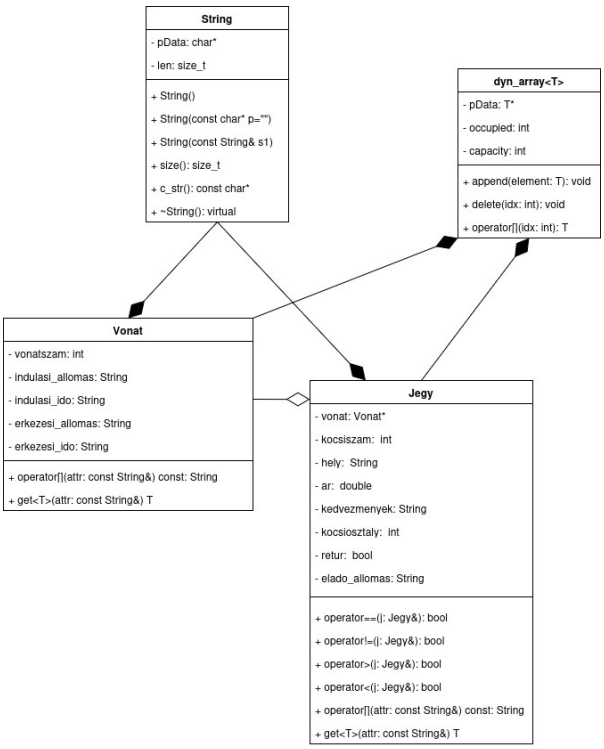
A csatolt osztálydiagramon nem minden tagfüggvény lett feltüntetve.

dyn_array<T>:
Ez egy dinamikus tömböt megvalósító sablon osztály. Három metódusa van:

- void append(T element); hozzáad egy elemet a tömb végéhez
- void delete(int idx); törli az idx indexű elemet
- T operator[](int idx); az idx indexű elemet teszi elérhetővé

Az append és delete hatására történő méret változásokat a tömb automatikusan kezeli. A lefoglalt terület méretét minden esetben a kétszeresére növeli ha több helyre van szükség. Törlésnél pedig akkor csökkenti a foglalt terület méretét a háromnegyedére ha az addig foglalt területnek kevesebb mint a fele van ténylegesen adattárolásra használva. Mindezt az occupied és a capacity mezők segítségével követi nyomon. Törlésnél az n-edik elem törlése után az n+1-edik lesz az n-edik, n+2-edik lesz az n+1-edik stb.

A csatolt osztálydiagramon konstruktorok nem lettek feltüntetve. Az osztály rendelkezik egy alapértelmezetten egy hosszúságú tömböt létrehozó konstruktorral amelynek megadható, hogy milyen hosszú tömböt szeretnénk.



Menüszerkezet:

A program parancssorból használható menürendszer által biztosítja az ígért funkciókat. Indítás után a főmenüben az alábbi opciók érhetőek el:

1. Vonat adatbázis betöltése/mentése
2. Jegy adatbázis betöltése/mentése
3. Vonat felvétele
4. Vonat törlése
5. Jegy kiadása
6. Menetrend generálása
7. Eladások összesítése
8. Kilépés

Ekkor a program nem tárol semmilyen információt de példa adatbázis (vonatokkal és jegyekkel) elérhető lesz. A megfelelő számú gomb megnyomása után az adott funkció hasonló menükben kéri be a folytatáshoz szükséges információkat.

Az első két lehetőség választásánál a fájlnév megadása után a felhasználó visszajelzést kap a művelet sikerességéről és a főmenü újra láthatóvá válik. A fájlnév megadása előtt dönteni kell arról, hogy az adatbázist betöltenénk vagy elmentenénk. Betöltésnél eldönthető, hogy a fájlból olvasott adatbázis a jelenleg tárolt mellé kerüljön a memóriában vagy felülírja azt. A program nem vizsgálja, hogy így egyezések létrejönnek-e.

Vonat felvételénél a *Vonat* osztály mezőinek értékét kell egyesével megadni majd itt is visszajelzést kapunk és a főmenü lesz újra látható. Ugyanígy történik a jegy kiadása is.

Vonat törlése esetén először kiválasztható, hogy a *Vonat* osztály mely mezője alapján szeretnénk törölni majd a találatokról egyesével eldönthető, hogy törölni szeretnénk-e vagy sem. Ekkor lehetőség van a művelet megszakítására vagy az összes találat törlésére is. A végén a program visszatér a főmenübe.

Menetrend generálásánál és Eladás összesítésnél a kimenet megjelenítésre kerül majd dönthetünk arról, hogy szeretnénk-e ezt fájlba menteni. Ha igen akkor meg kell adni a fájlnevet és utánna a főmenü lesz látható.

Készítette:
Pavlisinec Tamás