HÁZI FELADAT

Programozás alapjai 2.

Tervezés

Berényi Henrik Dániel
QP4TVJ

2021. április 11.

Tartalom

1. Feladat

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!

2. Feladatspecifikáció

A feladat egy vonatjegy eladó rendszer megvalósítása.

A feladat előírásának megfelelően megvalósítom a vonatok felvételét és a jegy kiadását. A feladat szövege nem írja, hogy hány jegyet, illetve vonatot kell tárolnia a programnak, így én amellett döntöttem, hogy dinamikus memóriakezeléssel tetszőleges számú vonatot lehet felvenni, illetve jegyet eladni.

Előírás, hogy a jegyekkel végezhető műveletek overloadolva legyenek, ezek közül a másolást, értékadást és kiírást valósítom meg.

A program menüvezérelt, a felhasználó a menüpontok számaival navigálhat a különböző menüpontok között.

A program indulásakor képes lesz egy fájlban előre eltárolt vonatok adatainak beolvasására, majd később a standard inputról való hozzáadásra is.
A program a vonatok és a jegyek hozzáadásakor az inputokat ellenőrzi és hibás adat megadása esetén(pl.: nem létező dátum, rossz járatszám formátum) kivételt dob.
A futást követően az új adatok a bemenetként használt fájlba mentődnek el.

A programhoz teszteseteket is készítek majd, amik helyes működés mellett a hibakezelést is ellenőrizni fogják.

3. Pontosított feladatspecifikáció

A feladat egy vonatjegy eladó rendszer megvalósítása.

A program képes vonatok és megvásárolt jegyek adatainak tárolására dinamikusan allokált memóriaterületen.

A vonatok és a jegyek adatai 1-1 .txt fájlban vannak eltárolva, amelyekből a program az indulásakor kiolvassa az összes információt és a futását követően eltárolja a változásokat is.

A szöveges fájlok adatainak formátuma a következő:
vonatok.txt:
azonosító#év#hónap#nap#óra#perc#év#hónap#nap#óra#perc#Indulási_állomás#Érkezési_állomás
jegyek.txt: név#kocsiszám#ülésszám#azonosító

Az adatok formátumai:
azonosító: a vonat azonosítója, 3 nagy betű és 3 szám ebben a sorrendben
év/hónap/nap/óra/perc: indulási és érkezési dátum azonosításához szükséges adatok, egész számok
indulási állomás/érkezési állomás/név: tetszőleges hosszú string (lehet benne szóköz)
kocsiszám/ülésszám: vonat kocsiját és az ülést azonosító számok

A program 6 osztállyal rendelkezik: a vonat osztály a vonatok adatait tárolja, a jegy osztály pedig a jegyek adatai mellett egy vonat objektumot is tartalmaz. Létezik egy Datum osztály is, amely tárolhatja egy járat indulási vagy érkezési időpontját. A szöveges adatokat a String osztály tárolja. Az utolsó két osztály heterogén tárolók, amelyek a Vonat és a Jegy objektumokat tárolják dinamikusan.

A Jegy osztályban az implicit függvények és a konstruktorok mellett a következő operátorok vannak definiálva:
- Értékadás(másoló konstruktorral együtt)
- Kiíró operátor(<<)

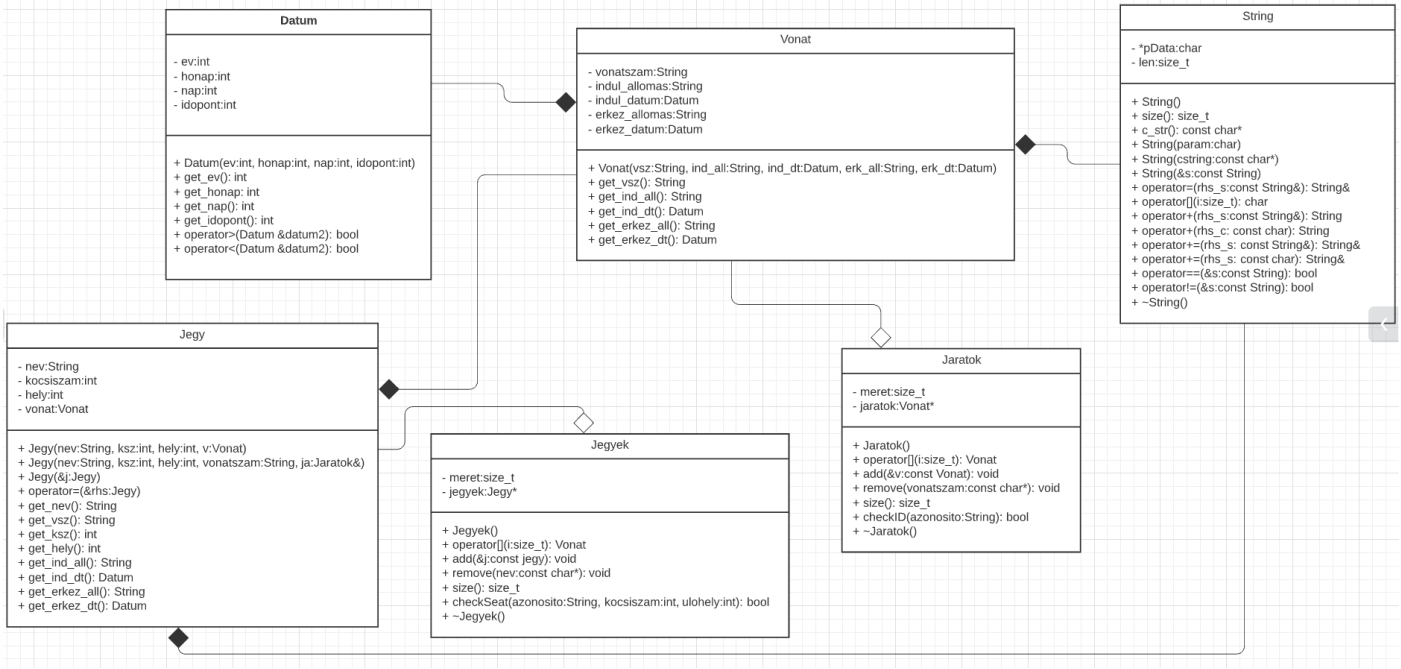
A program képes ellenőrizni a megadott inputok helyességét az azonosító, kocsi- és ülésszám és a dátumok esetében.
A program a fájlokból beolvasott adatokat nem ellenőrzi, azokat minden esetben helyesnek feltételezi, hiszen azt csak a program módosíthatja és íráskor csak a helyes adatokkal rendelkező objektumokat rögzíti.

4. Terv

A feladat az objektumok és a tesztprogram megtervezését igényli.

4.1. Objektum terv

Az objektumok részletes leírása a 3. fejezetben lévő pontosított specifikációban olvasható.



4.2. Algoritmusok

4.2.1. Jaratok osztály beolvas függvénye a fájlból való beolvasáshoz

```
void beolvas(Jaratok& j, const char* fajlnev) {
    ifstream fpp;
    fpp.open(fajlnev);
    if(!fpp) {
        printf( _Format: "KIVÉTEL\n");
        throw "QP4TVJ";
    }
    String* adatok = new String[13];

    int i = 0;
    while(getline( &: fpp, &: adatok[i], d: '#')) {
        if(i == 11) {
            getline( &: fpp,  &: adatok[12],  d: '\n');
            Datum ind( ev: stoi( &: adatok[1]),  honap: stoi( &: adatok[2]),  nap: stoi( &: adatok[3]),
                       idopont: stoi( &: adatok[4])*60+stoi( &: adatok[5]));

            Datum erk( ev: stoi( &: adatok[6]),  honap: stoi( &: adatok[7]),  nap: stoi( &: adatok[8]),
                       idopont: stoi( &: adatok[9])*60+stoi( &: adatok[10]));

            Vonat v( vsz: adatok[0],  ind_all: adatok[11], ind,  erk_all: adatok[12], erk);
            i = -1;
            j.add(v);
            delete[] adatok;
            adatok = new String[14];
        }

        i++;
    }
    delete[] adatok;
    fpp.close();
}
```

Ez a függvény beolvassa az eltárolt járatok adatait a megadott nevű fájlból, felhasználva a String osztályra definiált getline() és stoi() függvényeket. A getline() és stoi() függvények pontosan úgy működnek, mint az std::string hasonló nevű függvényei.

Beolvas függvény a Jegyek osztály esetén is létezik, azonban az a beolvasott adatokat kivéve nem különbözik az előbbitől.

4.2.2. Add függvény a heterogén kollekcióhoz való hozzáadáshoz.

```
void Jaratok::add(const Vonat& v) {
    Vonat* temp = new Vonat[meret+1];
    for(size_t i = 0; i < meret; i++) {
        temp[i] = jaratok[i];
    }
    temp[meret] = v;
    delete[] jaratok;
    jaratok = temp;
    meret++;
}
```

Add függvény a Jegyek osztály esetén is létezik, azonban az a beolvasott adatokat kivéve nem különbözik az előbbitől.

4.2.3. Remove függvény a heterogán kollekcióból való eltávolításhoz.

```
void Jaratok::remove(const char* vonatszam) {
    for(size_t i = 0; i < meret; i++) {
        if(jaratok[i].get_vsz() == vonatszam) {
            Vonat* temp = new Vonat[meret-1];
            size_t idx = 0;
            for(size_t j = 0; j < meret; j++) {
                if(jaratok[j].get_vsz() != vonatszam) {
                    temp[idx] = jaratok[j];
                    idx++;
                }
            }
            meret--;
            delete[] jaratok;
            jaratok = temp;
            return;
        }
    }
    throw "Nincs ilyen vonatszam!\n";
}
```

Remove függvény a Jegyek osztály esetén is létezik, azonban az a beolvasott adatokat kivéve nem különbözik az előbbitől.

4.2.4. Tesztprogram algoritmusai.

A programhoz készül majd egy tesztprogram is, amely a program összes függvényének működését teszteli a menü kivételével. A program ugyanis menüvezérelt és inputokat vár el a felhasználótól, majd a különböző menüpontok hívják meg a megfelelő függvényeket. A tesztprogrammal azonban sokkal előnyösebb egyenként tesztelni a függvényeket, megvizsgálni azok hibakezelését és működését.