Házi Feladat

Végleges

Fodor Dávid-D02DBR

1 Feladat:

- Készítsen filmeket nyilvántartó rendszert. Minden filmnek tároljuk a címét, lejátszási idejét és kiadási évét. A családi filmek esetében korhatár is van, a dokumentumfilmek esetében egy szöveges leírást is tárolunk. Tervezzen könnyen bővíthető objektummodellt a feladathoz! Demonstrálja a működést külön modulként fordított tesztprogrammal! A megoldáshoz ne használjon STL tárolót!

2 Feladatspecifikáció

- A feladat egy menü vezérelt filmtároló egység működését valósítja meg. A feladatban elvárt alapvető adatokat egy fájlban tárolja és abból dolgozik. Nem specifikálja a feladat, hogy esetleg sorozatokat is képes legyen a program tárolni és azok adatait is (évadok száma, kezdés éve, befejezés éve), de a program képes kell legyen erre is.
- A feladatban egy Film főosztállyal dolgozunk és a különböző műfajok ebből vannak származtatva. Heterogén kollekcióban tárolja őket a program, ami lehetővé teszi a különböző típusú elemek tárolását. És könnyen bővíthetőségét.
- A feladat nem adja meg külön, de a programnak képesnek kell lennie a következő alap műveletekre:
    - hozzáadás: új elem hozzáadása a kollekcióhoz
    - törlés: elem eltávolítása a kollekcióból
    - keresés: elem keresése a kollekcióban
    - kiírás: összes elem kiírása a képernyőre

- Az alapvető elvárt bemenetek egy fájlban vannak eltárolva, ezek típusai pl.:
    - cím (string)
    - lejátszási idő (int)
    - kiadási év (int)
    - korhatár (int) - csak családi filmek esetében
    - szöveges leírás (string) - csak dokumentumfilmek esetében
    - Csak ezeket az adattípusokat fogadja el a program a kért bemenetekkor, ellenkező esetben hiba lép fel
- Az elvárt kimenetek:
    - A program konzolos felületre írja és listázza ki az elvárt kimeneteket, de fájlba is elmenti törlés vagy hozzáadás esetén, így a következő futtatáskor tud vele dolgozni
- A program helytelen adat esetén hibajelzést dob
- Tesztelés
    - Tesztprogramnak a főprogramot fogom megvalósítani, amiben lesz példa elem hozzáadására, törlésére, és lesz benne hibás bemenet is, ami után a programnak hibát kell dobnia. A tesztprogram többféle műfajjal fog majd dolgozni, így látható lesz, a program működik minden alosztályra is.

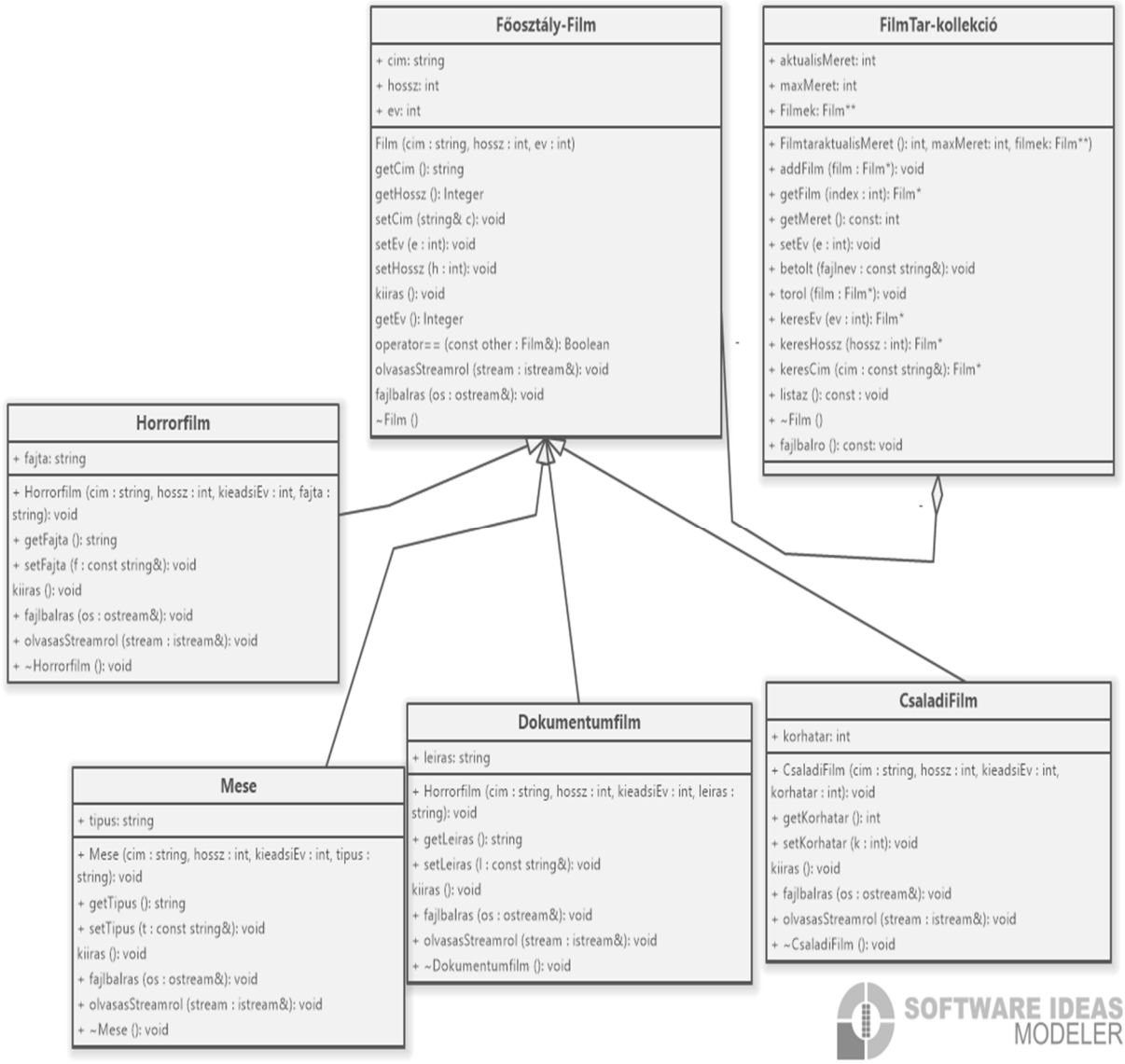
3 Terv

- Szükségem van a feladat megoldásához 6 osztályra legalább. Ebből egy a Film főosztály, amelynek leszármazottjai a különböző film műfajok, amik külön osztályok. Van egy FilmTar osztály, ami maga a kollekció, ez Film osztály pointereket tárol. Ez az osztály szolgál a fájlból olvasásra (egy darab fájlban vannak tárolva az adatok), a kiírásra, keresésre, törlésre és hozzáadásra a kollekcióhoz.
- A beolvasás úgy működik, hogy kap egy fájlt istream-ként az osztályok saját olvasó függvényei, majd ebben a főosztály olvasóját is felhasználva beállítja az értékeket a

setter függvényekkel, majd ezeket betölti egy objektumba és ezeket tárolja el a kollekció.

- Keresésre igazat ad, ha benne van a kollekcióban a keresett elem és hamisat, ha nincsen
- Törlésnél kitöri az adott elemet a kollekcióból, majd az új kollekciót visszaírja a fájlba, miután azt kiürítette.
- Hozzáadásnál, csak egy új elemet kap a kollekció

3.1 Osztálydiagram



4 Legfontosabb algoritmusok leírása

- A programban 4 nagyobb algoritmus van, ezek: betöltés, fájlba kiírás, kollekcióhoz adás és a törlés, ebben a részben ezeknek a működését fogom leírni

- Betöltés:
    - A függvény átvesz egy fájlnevet, majd ebből a fájlból létrehoz egy ifstream objektumot. Megvizsgálja, hogy sikerül-e megnyitni a fájlt, ha nem hibát ír ki és kilép a függvényből. Ezután amíg tud olvasni sort fut a ciklus és olvassa be a sorokat, amikből istringstream objektumot készít, majd vesszőig olvas be egy típust, és ennek megfelelően lép be a feltételekbe. Az adott típusból létrehoz egy objektumot, amiből csinál egy olyan típusú objektumot, majd ezt default konstruktorral inicializálja. majd ennek az objektumnak használja az olvasasStreamrol függvényét. Ez az ősosztály olvasasStreamrol virtuális függvényét használja, ami streamre olvassa a saját adatait, majd az alosztály ezen függvényei beállítják a saját plusz attribútumaikat is a streamre, majd ezek után a betolt függvény meghívja az adott elemre az addFilm függvényt.
- Fájlba kiírás:
    - Ezt is a FilmTar osztály függvénye valósítja meg, paraméter nélküli függvény, mely megnyitja a „Filme.txt” fájlt és egy ofstream objektumot csinál belőle, közben ki is törli a tartalmát. Ezután ellenőrzi, hogy megnyílt- e a fájl, ha nem runtime error hibát dob, amit el kell majd kapni. Ezután végig iterál a kollekción az aktuális méretig, és minden elemre meghívja a saját fájlba író függvényét. Minden alosztály kiírja a fájlba a saját fajtáját, majd meghívja az

ősosztály függvényét, amely a közös attribútumokról gondoskodik. Utána még a plusz attribútumát kiírja az alosztály függvénye, majd visszatér és bezáródik a fájl.

- Kollekcióhoz adás:
    - A FilmTar osztály függvénye átvesz egy Film típusú objektumot, hisz ezeket tárolja a kollekció, majd végig iterál a tömbön, hogy van e egyezés bármelyik másik elemmel, ha van kitörli ezt az elemet és kilép a függvényből. Utána, ha nincs ilyen, megnézi hogy az aktuális méret megegyezik e a maximum mérettel, mert, ha igen megnöveli a maximum méretet tízzel, hogy minél kevesebbszer kelljen másolgatni a tömb elemeit. Ezután ebbe az új tömbbe (ujFilmek) átmásolja a Filmake tömb elmeit. Majd törli a régi tömböt, és az újra állítja a pointert, és beállítja a maximum méretet az új méretre (max+10). Ha nem volt egyező a maximum méret és az aktuális méret, akkor csak az aktuális méret+1-edik helyre beteszi az új elemet és növeli egyel a méretet.
- Törlés:
    - Átvesz paraméterül egy Film pointert és végig iterál a kollekción, hogy ez benne van-e, ha igen, kitörli az elemet majd a helyén állít egy nullptr-t. Utána végig iterál a tömb utolsó előtti eleméig és minden helyre az utána lévő elemet állítja be. Ezután beállítja az utolsó elemet nullra, hisz itt már nincs adat. és csökkenti egyel a méretet.

5 Hibakezelés

- A programban néhány helyen van std::exception osztályból származó hiba, ahol kell, hogy láthassuk és le tudjuk kezelni az adott helyen mi történik.

- A getFilm függvényben, ha valaki az aktuális méretnél nagyobbat ad indexnek, akkor std::out_of_range hibát dob. Viszont, ha nulla alatti számot kap, ezt a számot átállítja nullára, hisz az lehet csak a legkisebb számú index.
- Azokban a függvényekben, ahol fájlt kell nyitni, ha nem sikerül a fájlt valami miatt, akkor itt a program std::runtime_error hibát fog dobniés kiírja, hogy hiba a fájl megnyitása közben.
- Ezeket a hibákat a tesztprogramban el is kapja a program és a .what() függvénnyel meg is nézi mi volt a baj.

6 Tesztelés

- A program kettő tesztprogramot futtat, egy általam megírt tesztesetekkel, amiben minden lehetőségre van egy teszt, hogy látható legyen a működés, és van benne hibás adat is, hogy látható legyen, hogy a hibakezelési felület is működik és ezzel a coverage tesztben is látható legyen a kihasználtsága.
- Ebben a tesztesetben minden lehetőségre lesz példa, ahol nem jó a bemenet, vagy olyan bemenet van, amit nem találhat meg a program. Van egy hozzáadás, amiben a film címe „alma”, hogy látszódjon teljesen, hogy működik a hozzáadás.
- A második teszteset ugyan úgy működik, mint az első teszteset, ugyan azt csinálja, ugyan abban a sorrendben, csak ebben az esetben a standard inputról olvas be adatokat, melyeket egy erre alkalmas fájlban tárolok, hogy a JPorta tesztelése ezekkel az adatokkal tudja tesztelni. Ezek az adatok a Szabvanyos_bemenet.txt fájlban vannak eltárolva és lesznek feltöltve a JPorta rendszerébe.

7 Önrevízió

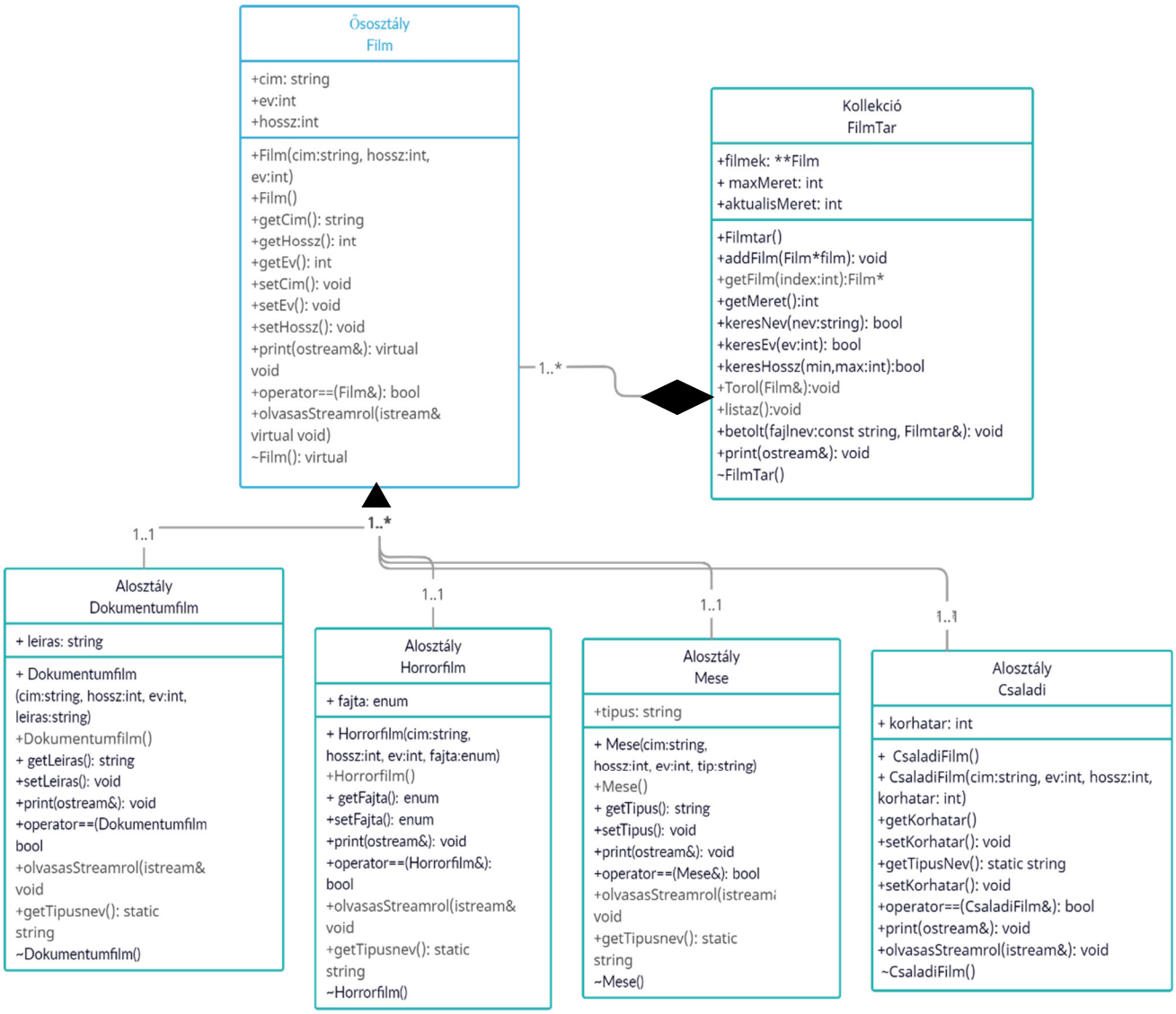
- A feladat elkészítése alatt sok dolog változott az alapvető elképzeléshez képest és az első tervekhez képest is, ebben a részben foglalom össze, hogy mik is ezek a változások.
- A program nem dolgozik pluszban sorozatokkal, hisz a heterogén kollekcióban tárolásnál számba kellett vegyem, hogy túl sok olyan tulajdonsága van, ami nincs meg a filmekben (pl. epizódok, évadok, rész hosszok, stb.)
- A program nem menü vezérelt program, hisz a feladat elvárása nem adta ezt meg, így csak egy tesztprogram van, mely interaktív felületet biztosít a felhasználó számára.
- Keresésre a keres… függvények nem bool visszatérési értékkel dolgoznak, hanem egy film pointert, tehát az adott elemre mutató pointert adnak vissza, és nullptr-t ha nincs benne, vagyis nincs egyezés a tömbben, ez azt is segíti, hogy ne csak azt tudhassuk meg, hogy benne van-e a tömbben az adott elem, hanem abban is, hogy ezzel az elemmel lehessen dolgozni utána is.
- A törlés nem törli egyből a fájlból az elemet, hanem kitörli az ad□tt elemet amit, a getFilm függvény segítségével ér el és azt a k□llekcióból törli, majd minden végén a fájlbaíró függvénnyel kerül vissza a fájlba a megfelelő f□rmátumba.

8 Önrevízió, osztáyokban történt változások

- Ebben a részben leírom mik változtak az osztályokban a terv rész óta, ehhez a végére csatolom az akkor készült osztálydiagramot.
- Film osztály:
    - bekerült a virtual void olvasasStreamrol(std::istream& stream); függvény

    - bekerült a virtual void fajlbaIras(std::ostream& os) const ; függvény
- Dokumentumfilm osztály:
    - eltűnt a getTipusnev függvény, mivel már nem templates a fájlból betöltés, mert azzal csak egyféle elemre működne jól egy beolvasás, de nekem az összesre kell, így ez a függvény csak ahhoz kellett, ezért már nincs rá szükség
    - bekerült a void kiiras() const override; függvény
    - bekerült a void fajlbaIras(std::ostream& os) const; függvény
- Mese osztály
    - kikerült az operator== felüldefiniálás, mivel csak Film objektumot kell összehasonlítani a program használata alatt, azt is csak ellenőrzés céljából, Mese objektumot viszont nem kell, így ez feleslegessé vált
    - eltűnt a getTipusnev függvény, mivel már nem templates a fájlból betöltés, mert azzal csak egyféle elemre működne jól egy beolvasás, de nekem az összesre kell, így ez a függvény csak ahhoz kellett, ezért már nincs rá szükség
    - bekerült a void kiiras() const override; függvény
    - bekerült a void fajlbaIras(std::ostream& os) const; függvény
- Horrorfilm osztály
    - kikerült az operator== felüldefiniálás, mivel csak Film objektumot kell összehasonlítani a program használata alatt, azt is csak ellenőrzés céljából, Mese objektumot viszont nem kell, így ez feleslegessé vált
    - eltűnt a getTipusnev függvény, mivel már nem templates a fájlból betöltés, mert azzal csak egyféle elemre működne jól egy beolvasás, de nekem az összesre kell, így ez a függvény csak ahhoz kellett, ezért már nincs rá szükség
    - bekerült a void kiiras() const override; függvény

    - bekerült a void fajlbaIras(std::ostream& os) const; függvény
    - az fajtája tehát a saját adata a horrorfilm osztálynak enum helyett string lett, a horrorfilmek fajtáinak sokszínűsége miatt egyszerűbb csak beolvasni, mint minden név opciót feljegyezni az enumban
- CsaladiFilm osztály
    - kikerült az operator== felüldefiniálás, mivel csak Film objektumot kell összehasonlítani a program használata alatt, azt is csak ellenőrzés céljából, Mese objektumot viszont nem kell, így ez feleslegessé vált
    - eltűnt a getTipusnev függvény, mivel már nem templates a fájlból betöltés, mert azzal csak egyféle elemre működne jól egy beolvasás, de nekem az összesre kell, így ez a függvény csak ahhoz kellett, ezért már nincs rá szükség
    - bekerült a void kiiras() const override; függvény
    - bekerült a void fajlbaIras(std::ostream& os) const; függvény
- FilmTar osztály (heterogén kollekció)
    - A betolt függvény nem kap filmtar referenciát, csak egy fájlnevet
    - a print függvény neve fajlbaIro
    - a keres… függvények visszatérési értéke nem bool hanem maga az elem, amit megtalált
    - a kereshossz nem egy értéktartományt kap paraméterül (min.-max.), mert lehet egy tartományba beleférne az összes film, ekkor ennek a keresésnek nem lenne értelme
- Régi diagram:


Ősosztály
Film
+cim: string
+ev:int
+hossz:int
+Film(cim:string, hossz:int, ev:int)
+Film()
+getCim(): string
+getHossz(): int
+getEv(): int
+setCim(): void
+setEv(): void
+setHossz(): void
+print(ostream&): virtual void
+operator==(Film&): bool
+olvasasStreamrol(istream& virtual void)
~Film(): virtual
Kollekció
FilmTar
+filmek: **Film
+ maxMeret: int
+aktualisMeret: int
+Filmtar()
+addFilm(Film*film): void
+getFilm(index:int):Film*
+getMeret():int
+keresNev(nev:string): bool
+keresEv(ev:int): bool
+keresHossz(min,max:int):bool
+Torol(Film&):void
+listaz():void
+betolt(fajlnev:const string, Filmtar&): void
+print(ostream&): void
~FilmTar()
1..*
1..1
1..*
Alosztály
Dokumentumfilm
+ leiras: string
+ Dokumentumfilm (cim:string, hossz:int, ev:int, leiras:string)
+Dokumentumfilm()
+ getLeiras(): string
+setLeiras(): void
+print(ostream&): void
+operator==(Dokumentumfilm bool
+olvasasStreamrol(istream& void
+getTipusnev(): static string
~Dokumentumfilm()
1..1
Alosztály
Horrorfilm
+ fajta: enum
+ Horrorfilm(cim:string, hossz:int, ev:int, fajta:enum)
+Horrorfilm()
+ getFajta(): enum
+setFajta(): enum
+print(ostream&): void
+operator==(Horrorfilm&): bool
+olvasasStreamrol(istream& void
+getTipusnev(): static string
~Horrorfilm()
1..1
Alosztály
Mese
+tipus: string
+ Mese(cim:string, hossz:int, ev:int, tip:string)
+Mese()
+ getTipus(): string
+setTipus(): void
+print(ostream&): void
+operator==(Mese&): bool
+olvasasStreamrol(istream& void
+getTipusnev(): static string
~Mese()
1..1
Alosztály
Csaladi
+ korhatar: int
+ CsaladiFilm()
+ CsaladiFilm(cim:string, ev:int, hossz:int, korhatar: int)
+getKorhatar()
+setKorhatar(): void
+getTipusNev(): static string
+setKorhatar(): void
+operator==(CsaladiFilm&): bool
+print(ostream&): void
+olvasasStreamrol(istream&): void
~CsaladiFilm()