HÁZI FELADAT

Programozás alapjai 2.

Feladatválasztás & Feladatspecifikáció

Dancs Krisztián

2022.március.28.

Tartalom

1. Feladat

Filmpédia:

Készítsen filmeket nyilvántartó rendszert. Minden filmnek tároljuk a címét, lejátszási idejét, kiadási évét illetve rendezőjét. A filmekhez lehessen adni egy rövid leírást is. Tervezzen könnyen bővíthető objektummodellt a feladathoz! Demonstrálja a működést külön modulként fordított tesztprogrammal! A megoldáshoz ne használjon STL tárolót!

2. Feladatspecifikáció

A feladat egy filmek tárolására alkalmas filmpédia elkészítése. Az általam készített String osztályt fogom használni STL tároló használata helyett.

A filmeket fájlba lehet menteni, illetve kiolvasni azokat. Filmek összeadása operátortúlterheléssel lesz megvalósítva, amelyet egy Maraton nevű függvény fog használni. A Maraton függvény paraméterként több filmet fog kapni, amelyeknek megkapjuk az összhosszát, órában. Ezzel ellentétben a filmek lejátszási ideje percben lesz megadva.

Minden filmnek lesz egy:

- Neve/címe (String)
- Rendezője (String)
- Lejátszási hossza (Integer)
- Kiadási éve (Integer)
- Rövid leírás (String)

A programhoz fog tartozni egy menü is amely három alponttal fog rendelkezni:

- Film létrehozása
- Filmek listázása
- Film törlése

Minden filmet és az adatait egy .csv fájlban fogjuk tárolni és kiolvasni. Hibás fájl esetén kivételt dob a program. Ha a felhasználó szeretne készíteni egy Maratont a meglévő filmjeiből, akkor a Maraton függvény több paramétert fogad el és ezeknek összeadja a lejátszási hosszát és visszatér azzal, így a felhasználó tudni fogja, hogy mennyi időt venne fel az adott számú filmeket egymás után sorba megnézni.

Négy fajta filmet fogunk megkülönböztetni:

- Hollywood
- Művészfilm
- Európai mozifilm
- Távol-keleti mozifilm

Az adott filmekhez fognak tartozni egyedi tulajdonságok is. A filmek felvételéhez tartozó hibás inputokat a program kivételkezeléssel fog hibaüzeneteket visszadobni.

A teszteléshez, egy olyan programot készítek ami különböző inputokkal teszteli a megírt funkciókat és az azok által dobott illetve elvárt kivételeket.

3. Pontosított Feladatspecifikáció

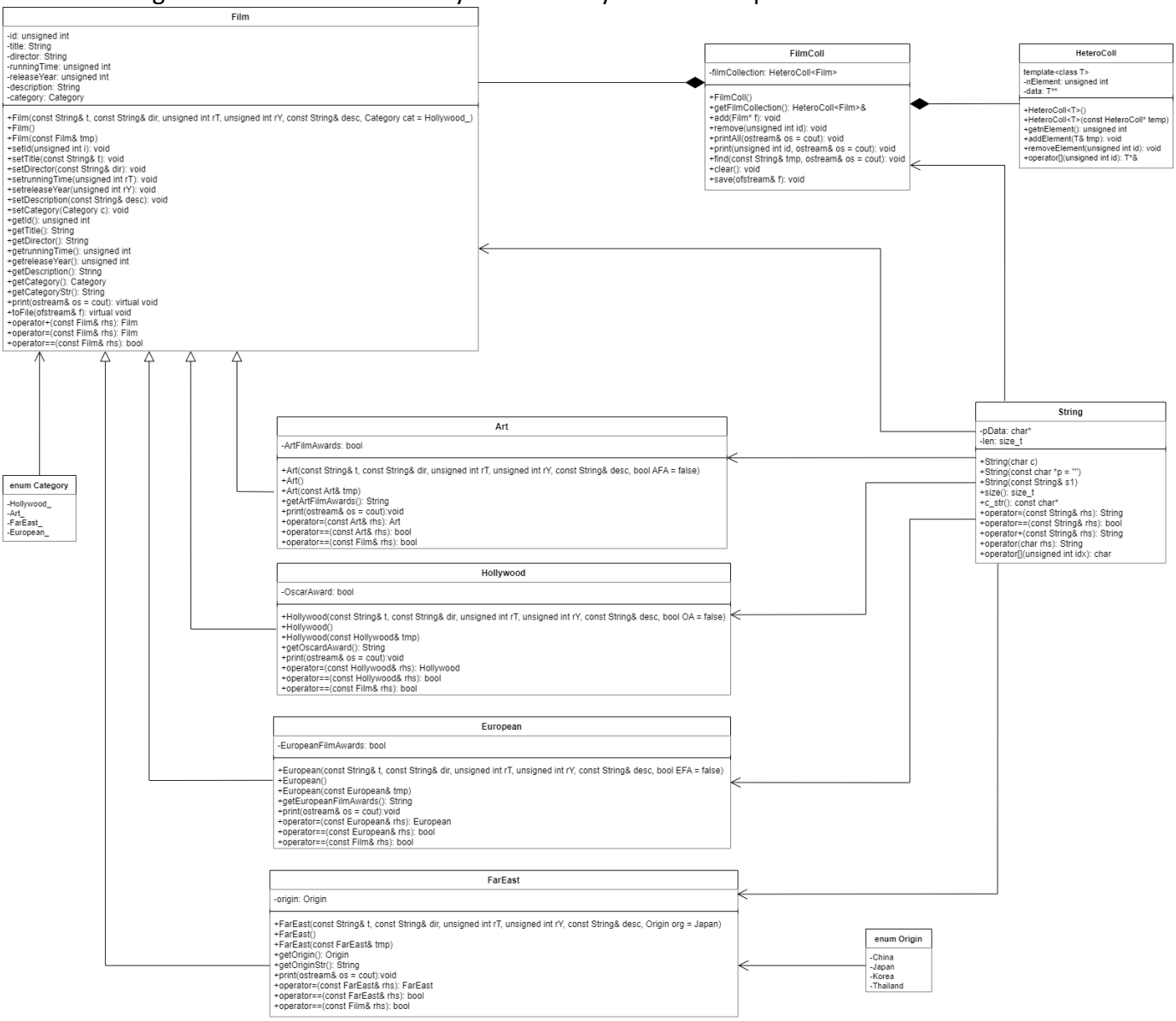
A megkülönböztetett filmkategóriák örökölni fogják a Film osztályt, ezzel főbb tulajdonságukat megörökölve. Emiatt, egy film fontosabb függvényei és hozzá tartozó attribútumai a Film osztályban lesznek megírva. Az osztályoknak, ezen kívül meg lesz írva minden fontosabb függvénye illetve operátora is pl: kiirató operátor, fájlbaírás stb.

A filmek az alábbi egyedi attribútummal fognak rendelkezni:

- Hollywood: Nyert-e Oscar díjat? (boolean)
- Art: Nyert-e Art Film Awards díjat? (boolean)
- FarEast: Honnan ered a film? (China, Japan, Korea, Thailand) (enum)
- European: Nyert-e European Film Awards díjat? (boolean)

4. Terv

Az általam választott feladatot 7 osztály fogja leírni, ebben szerepel az általam megírt String osztály is, illetve a heterogén kollekciót kezelő osztály is. Az osztályok közötti kapcsolat itt látható:



5. Algoritmusok

A programban minden filmhez tartozni fog egy print illetve toFile függvény. Ez azért szükséges és azért nem lehet örökléssel megoldani, mivel különböző filmeknek különböző tulajdonságaik vannak. A print ki fogjak írni a film id-ját (a heterogén kollekcióban hányadik helyen helyezkedik el), a címét, a rendezőjét, a lejátszási idejét, kiadási évét, a hozzá tartozó leírást, a kategóriáját az adott filmnek illetve a saját tulajdonságát. Ezek mellett mindegyik Filmet öröklődő osztálynak két egyenlőség vizsgáló operátora lesz, az egyik amikor két hasonló típusú filmet hasonlít össze, illetve lesz egy másik is ami mindig false-t fog visszaadni, ez abban az esetben kell ha két nem ugyan olyan típusú filmet akarunk összehasonlítani.

A program ezek mellett fog tesztelni és kivételt dobni, ha alul- vagy felülindexelés van, illetve ha más nem várandó inputokat kap a felhasználótól.

6. Fontosabb függvények

A programom főként a heterogén adatstruktúrára épít, ezért a legfontosabb függvényeim is a heterogén kollekció osztályomban vannak jelen.

void add(Film* f);

- Hozzáad a heterogén kollekcióhoz egy filmet
- Param: egy film pointerét kell megadni
- Return: nincs

void remove(unsigned int id);

- Töröl, tehát felszabadít egy adott filmet a kollekcióból, és csökkenti a kollekció elemszámát 1-el
- Param: egy film id-ja (unsigned int id)
- Return: nincs

void printAll(std::ostream& os = std::cout) const;

- Kiírja egy kollekció minden filmét egy adott ostream-re
- Param: ostream amire kiírja a filmeket (ostream& os)
- Return: nincs

void print(unsigned int id, std::ostream& os = std::cout) const;

- Kiírja egy adott id-jű film adatait egy adott ostreamre
- Param: - egy film id-ja (unsigned int id)
  - ostream amire kiírja az adatokat (ostream& os)
- Return: nincs

void find(const String& tmp, std::ostream& os = std::cout) const;

- Megkeres egy filmet, ha a film címe megegyezik a kereséssel, akkor kiírja azoknak az adatait
- Param: a keresett film címe (String) illetve egy ostream amire kiírat (ostream& os)

void clear();

- Törli az egész kollekciót és az elemszámot nullára állítja

- Param: nincs
- Return: nincs

A következő két függvény a legkomplexebb, mivel nem egyszerű fájlba iratást és fájlból kiolvasást alkalmaz, hanem egy adott film adatait még tovább kell parseolni különböző beépített illetve általam megírt függvények használatával (strtoul(), retCat(String) stb.).

void save(ofstream& f, String fajl);

- Paraméterként kapott fájlba menti a heterogén kollekció filmjeit
- Param: egy filestream (ofstream& f), illetve egy fajlnev (String)
- Return: nincs

void fromFile(String s);

- Paraméterként megadott nevű fájlból kiolvassa az adatokat és filmekbe rendezi, amiket aztán heterogén kollekcióhoz ad
- Param: egy fájl neve (String)
- Return: nincs

7 . Hibakezelés

A program főként a bemeneti hibákat kell kezelje amiket a felhasználó ad meg. Kevés try catch blokkot használ a program, mivel a menüben bármilyen hibába futunk azonnal kezelnünk is kell a felhasználó számára. Példa a hibakezelésre:

```
try {
    coll.find(s.c_str(), cout);
  } catch(char const* err) {
    cout << "Kivetelt dobott mivel nem letezik ilyen film" << endl;
  }
```

```
try {
    coll.remove(r);
    cout << "Film torulve!" << endl;
  } catch(char const* err) {
    cout << "Kivetelt dobott mivel nem letezik ilyen id" << endl;
  }
```

Ez a két try catch blokk azokat a bejövő inputokat kezeli amelyeket a felhasználó hibásan adhatott meg. A legtöbb input ha hibás akkor törlődnek és új lehetőséget ad a felhasználónak az adatok helyes megadására. Ilyenek például az olyan adatok mint a lejátszási idő illetve a kiadási év. Ezek nem lehetnek stringek, csakis unsigned int típusúak lehetnek, ezért a progam a bejövő inputot ami a lejátszási időhöz vagy a kiadási évhez tartozik, akkor a program karakterről karakterre leellenőrzi, és ha talál egy karaktert

akkor hibát illetve hibaüzenetet dob vissza. A memóriaszivárgást a program mindenhol lekezeli és felszabadítja.

8 . A program használata

A programot batch programként lehet futtatni, ahol esetleges buildelés (make all paranccsal) után egy menü fogadja a felhasználót, ahol választani tud különböző opciók közül. Az egyes opcióknál ahol bemenetet kell megadni, ajánlott a megadott utasítások szerint cselekedni, mivel ha nem jó bemenetet kap a program akkor kiír egy hibaüzenetet és a felhasználó újrapróbálhatja. A program végezte után ajánlatos a filmeket egy fájlba lementeni, különben elvesznek a kollekcióból, ezután már sikeresen ki lehet lépni a 0-ás opcióval.