HÁZI FELADAT

Programozás alapjai 2.

Kész

Tóth Mihály Balázs
OAYOF1

2024. 04. 24.

TARTALOM

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 filmeket tároló nyilvántartás létrehozása. Egy filmnek több attribútuma lehet (cím, játékidő, kiadás éve) a különböző típusú filmek egyedi attribútumai is eltárolhatók, például családi filmeknél a korhatár, dokumentumfilmeknél egy rövid leírás. A nyilvántartáshoz könnyen hozzá lehet adni új filmeket és új osztályok létrehozásával új filmtípusokat is.
Alapvető funkciója a nyilvántartásnak a kereshetőség, film címe alapján. Ha nem található az adott cím akkor hibát dob a rendszer. A katalógushoz hozzá lehet adni és ki lehet belőle törölni filmeket. Továbbá lehetőségünk van a teljes adatbázis kiírására, illetve beolvasására is .txt fájlból.
A filmek adatai a tetszőlegesen hosszú címet leszámítva fix méretű tárolókban vannak nyilvántartva, míg maguk a filmek dinamikusan foglalt területen, így könnyen bővíthető az új filmek hozzáadásával.
A tesztprogram egy előre elkészített adatbázist tölt be, ezen demonstrálja a keresést, a hozzáadást és a törlést.

3. Terv

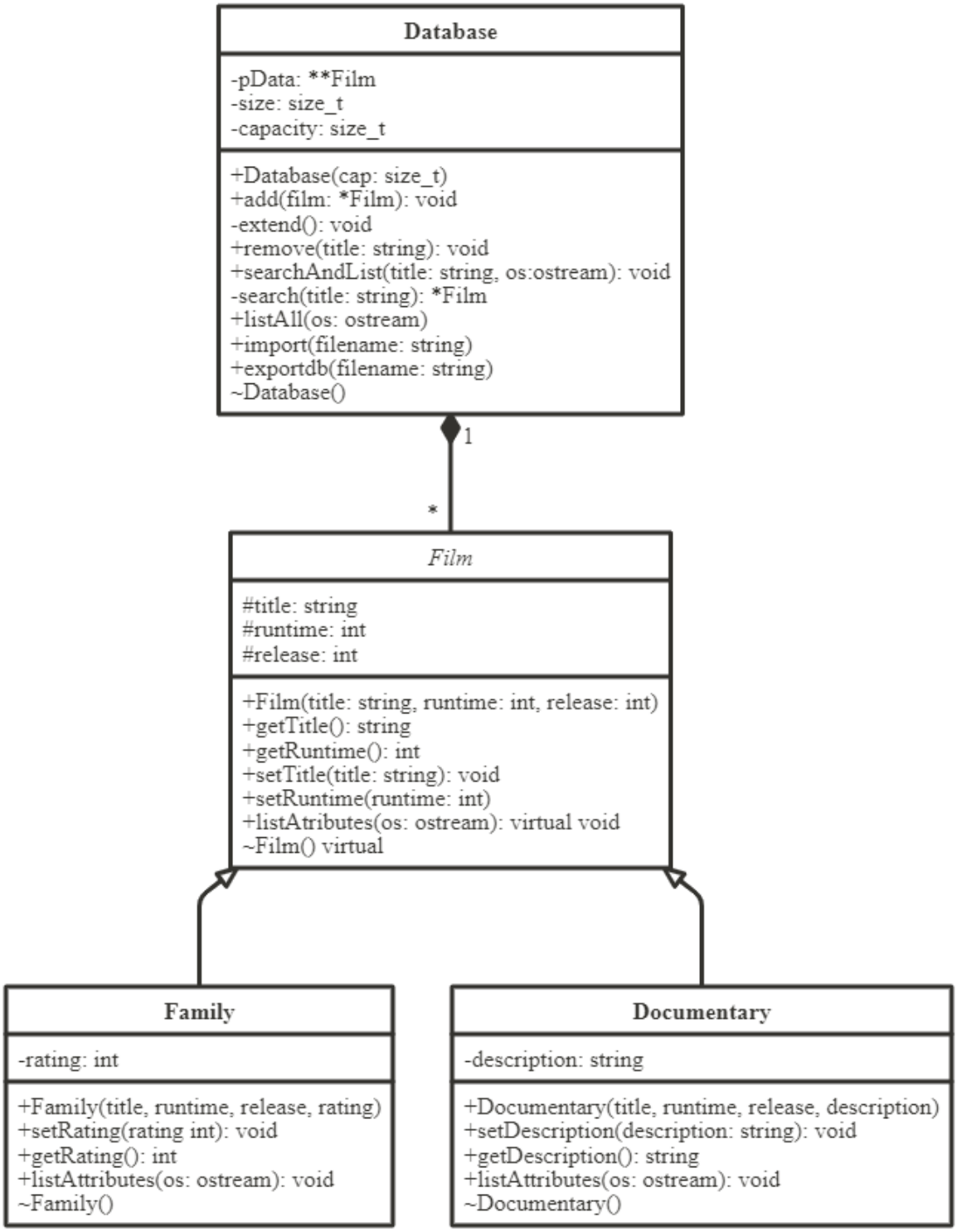
A feladat objektumok és a tesztprogram megtervezését igényli.

3.1. Objektum terv

A megvalósításhoz az alábbi UML diagramon látható 4 osztályt hozom létre:

- Datbase: Heterogén kollekció, Film pointereken keresztül tárolja a különféle Film objektumok példányait. Elsődleges feladata a dinamikus tároló karbantartása.
- Film: Absztrakt ősosztály, ebből példányosodik a Family és Documentary. Tárolja a címet, a játékidőt és a megjelenési évet.

- Family: Családi film alosztály, a Film-hez képest tárolja a korhatárt és felüldefiniálja a szükséges függvényeket
- Documentary: Dokumentumfilm alosztály, a Film-hez képest tárolja a leírást és felüldefiniálja a szükséges függvényeket



3.2. Algoritmusok

Alább a megoldáshoz szükséges algoritmusok közül csak a bonyolultabbakat említem.

3.2.1. Database :: add

Ezzel a függvénnyel lehet hozzáadni a tárolóhoz új filmet, a beillesztendő pointer használatával. Ellenőrzésre kerül, hogy van-e elég szabad hely a tárolóban (ha nincs, megnöveli) majd hozzáfűzi a tömbhöz.

3.2.1.1 Database :: extend

A tárolónak foglalt terület megnövelésére szolgál. A költséges memóriafoglalási és másolási műveletek számának minimalizálása érdekében minden hívásra foglal egy dupla akkora méretű tömböt és ebbe másolja át az elemeket, az eredetit felszabadítva.

3.2.2. Database :: remove

Megkeresi az adott címet és eltávolítja a tárolóból. A helyén keletkező lyuk feltöltéséhez az aktuális utolsó tömbelemet áthelyezi ide.
Ha nincs ilyen című elem, akkor *char const** kivételt dob „entry not found” értékkel.

3.2.3. Database :: searchAndList

A kapott cím alapján megkeresi majd listázza az elem attribútumait a paraméterül kapott streamre, az adott *listAttributes* függvényének segítségével
Ha nincs találat, a fent említett kivételt dobja.

3.2.4. Database :: import

A paraméterül kapott fájlból beolvassa a rekordokat és felépíti a tárolót. Egy film adatai egy sorban vannak tárolva tabulátorokkal elválasztva. A sorokat beolvassa, és az ennek megfelelően létrejövő objektumokat elhelyezi az adatbázisban.
Olvasási hiba esetén *char const** kivételt dob „file not found” értékkel.

3.2.5. Database :: exportdb[1]

A megadott nevű fájlba exportálja az adatbázist, olyan formában, hogy az *import* függvény azt később kezelni tudja.
Megvalósításához a *listAll* függvény kimenetét irányítja át.
Megnyítási hiba esetén szintén *char const** kivételt dob „file not found” értékkel.

3.2.6.Tesztprogram algoritmusai[2]

- A tesztprogram a mellékelt adatbázisból file végéig olvas, ezzel betöltve a tárolót.
- Ezután megpróbálja listázni egy nem létező film adatait.
- Hozzáadja a filmet.
- Majd újra meghívja rá az adatokat listázó eljárást.

[1] „export”-ról átnevezve név ütközés miatt.
[2] A jelenleg implementált tesztek nem ezt a tervet tükrözik, a végleges tesztek is eltérhetnek.

- Listázza az összes filmet.
- Törli az egyiket.
- Végül megpróbálja a most törölt film adatait listázni.

4. Megvalósítás

4.1. Osztályok és függvényeik

4.1.1. Database

A heterogén kollekció, a filmek tárolója.

4.1.1.1 Változók

Név	Típus	Funkció
pData	Film**	Film pointerek tárolója
size	size_t	A tároló aktuális mérete
capacity	size_t	A tároló kapacitása

4.1.1.2 Függvények

Database::Database(size_t cap)
Adatbázis konstruktor, létrehozza az adatbázist a megadott kezdeti kapacitással.
Alapértelmezett kapacitás megadása a defCap makróval.

Database::~Database()
Adatbázis destruktor, feladata a dinamikusan lefoglalt erőforrások felszabadítása. Felszabadítja a teljes tartalmat, majd megszünteti az üres adatbázist.

void Database::add(Film *film)
Film hozzáadása az adatbázishoz.
A függvény hozzáadja a pointerrel paraméterül kapott Film objektumot az adatbázishoz. Ha az adatbázis megtelik, megnöveli a tároló méretét.

void Database::extend()
Tároló méretének növelése
Ez a függvény akkor hívódik meg, ha az adatbázis megtelik, és új elemet kell hozzáadni, de nincs elég hely a tárolóban.
A függvény létrehoz egy új, dupla méretű tárolót, átmásolja bele az eddigi elemeket, majd felszabadítja az eredeti tárolót.

Film *Database::search(std::string title)
Ez a függvény végigiterál az adatbázis tárolóján, és megkeresi a paraméterként kapott címmel rendelkező filmet.
Ha megtalálja, visszaadja annak a pointer-ét, különben kivételt dob az "entry not found" üzenettel.

void Database::remove(std::string title)
Film eltávolítása az adatbázisból

A függvény eltávolítja az adatbázisból a paraméterként kapott címmel rendelkező filmet.
Ha megtalálja a filmet, akkor felszabadítja a memóriát, majd az utolsó elemmel felülírja a törlendőt, és csökkenti a tároló méretét, ezzel megszüntetve a tömbben keletkező lyukat.
Ha nem találja meg a filmet akkor kivételt dob az "entry not found" üzenettel.

void Database::searchAndList(std::string title, std::ostream &os)
Film keresése és adatainak listázása
Ez a függvény megkeresi a paraméterként kapott címmel rendelkező filmet az adatbázisban, majd kiírja annak adatait a megadott ostream-re.

void Database::listAll(std::ostream &os)
Az összes film adatainak listázása
Ez a függvény kiírja az adatbázisban tárolt összes film adatait a megadott ostream-re.

void Database::import(std::string filename)
Adatbázis importálása fájlból
Ez a függvény betölti az adatbázisba a megadott fájlban tárolt filmek adatait.
A fájlban a filmek típusa után a megfelelő adatok szerepelnek, a függvény a film típusának megfelelően tölti be az értékeket.
Ha a fájl megnyitása sikertelen a „file not found” errort dobja.

void Database::exportdb(std::string filename)
Adatbázis exportálása fájlba
Ez a függvény kiírja az adatbázisban tárolt összes film adatait a megadott fájlba.
Ha a fájl megnyitása sikertelen a „file not found” errort dobja.

4.1.2. Film

Absztrakt ősosztálya a különféle filmtípusoknak.

4.1.2.1 Változók

Név	Típus	Funkció
title	std::string	A film címe
runtime	int	Játékidő (perc)
release	int	Kiadási év

4.1.2.2 Függvények

Film::Film(std::string title, int runtime, int release)
Film osztály konstruktora
Létrehozza a Film objektumot a megadott címmel, játékidővel és kiadási évvel.

std::string Film::getTitle()
Visszaadja a film címét.

int Film::getRuntime()
Visszaadja a film játékidejét.

int Film::getRelease()
Visszaadja a film kiadási évét.

void Film::setTitle(int title)
Beállítja a film címét a megadott stringre.

void Film::setRuntime(int runtime)
Beállítja a film játékidejét a megadott értékre (percben).

void Film::listAttributes(std::ostream &os)
Kiírja a film attribútumait a megadott ostream-re.
Virtuális függvény.

4.1.3. Family

A Film osztályból származtatott családi film osztály.

4.1.3.1 Változók

Örökli a Film osztály adattagjait, illetve

Név	Típus	Funkció
rating	int	Korhatár

4.1.3.2 Függvények

Family(std::string title, int runtime, int release, int rating)
Family osztály konstruktora
Létrehozza a Family objektumot a megadott címmel, játékidővel, kiadási évvel és korhatárral.
Ehhez a Film osztály konstruktorát veszi igénybe.

void Family::setRating(int rating)
Beállítja a film korhatárát a megadott értékre.

int Family::getRating()
Visszaadja a film korhatárát.

void Family::listAttributes(std::ostream &os)
Kiírja az objektum attribútumait a megadott ostream-re.
A Film osztály azonos nevű függvényének override-ja.

4.1.4. Documentary

A Film osztályból származtatott dokumentumfilm osztály.

4.1.4.1 Változók

Örökli a Film osztály adattagjait, illetve

Név	Típus	Funkció
description	std::string	Rövid leírás a filmhez

4.1.4.2 Függvények

Documentary(std:: string title, int runtime, int release, std:: string desc)
Documentary osztály konstruktora
Létrehozza a Documentary objektumot a megadott címmel, játékidővel, kiadási évvel és leírással.
Ehhez a Film osztály konstruktorát használja fel.

void Documentary::setDescription(std::string desc)
Beállítja a dokumentumfilm leírását a megadott stringre.

std::string Documentary::getDescription()
Visszaadja a film leírását.

void Documentary::listAttributes(std::ostream &os)
Kiírja az objektum attribútumait a megadott ostream-re.
A Film osztály azonos nevű függvényének override-ja.

4.2. Tesztprogram

A program több unit testet tartalmaz az implementált osztályok és funkciók tesztelésére. A tesztesetek kiválasztásához a laborfeladatokhoz hasonlóan egy a fájl elején lévő makrót használ.
A tervhez képest lényegi változás, hogy az egyes funkciók a hibafeltárás megkönnyítésére először egymástól izolálva vannak tesztelve, illetve nem használ csatolt adatbázist, hanem magának exportál egyet az import testhez

A unit testek az osztáylok alapvető működésének ellenőrzésével kezdődnek, ezek a következők:

- FamilyTest: Ez a teszteset a Family osztály működését vizsgálja. Létrehoz egy Family objektumot a megadott címmel, játékidővel, kiadási évvel és korhatárral, majd ellenőrzi, hogy a getter függvények helyesen működnek-e, és visszaadják-e a megfelelő értékeket.

- DocumentaryTest: Ez a teszteset a Documentary osztály működését teszteli. Hasonlóan a FamilyTest-hez, létrehoz egy Documentary objektumot a megadott

címmel, játékidővel, kiadási évvel és leírással, majd ellenőrzi, hogy a getter függvények helyesen működnek-e, és visszaadják-e a megfelelő értékeket.

- DatabaseTest: Ez a teszteset a Database osztály működését ellenőrzi. Létrehoz egy üres adatbázist, hozzáad néhány Family és Documentary objektumot, majd ellenőrzi, hogy a listAll függvény helyesen kiírja-e az összes film adatait.

A további tesztesetek már a program magasabb funkcióit vizsgálják

- DatabaseSearchTest: Ez a teszteset az adatbázis keresési funkcióját vizsgálja. Hozzáad néhány Family és Documentary objektumot az adatbázishoz, majd keres egy adott című filmet. Ellenőrzi, hogy a keresés eredménye megfelel-e az elvárt eredménynek.

- DatabaseExportTest: Ez a teszteset az adatbázis exportálási funkcióját teszteli. Létrehoz egy adatbázist, hozzáad néhány filmet, majd exportálja az adatokat egy fájlba. Ellenőrzi, hogy az exportált fájl tartalma megfelel-e az elvárt eredménynek.

- DatabaseImportTest: Ez a teszteset az adatbázis importálási funkcióját ellenőrzi. Importálja az előbb elkészített tesztfájlt az adatbázisba, majd ellenőrzi, hogy az importált adatok megfelelően lettek-e feldolgozva és betöltve.

- DatabaseRemoveTest: Ez a teszteset az adatbázisból törlés funkcióját teszteli. Hozzáad néhány Family és Documentary objektumot az adatbázishoz, majd töröl egy adott filmet. Ellenőrzi, hogy a törlés után az adatbázis tartalma megfelel-e az elvárt eredménynek.

Az utolsó tesztek a hibakezelést ellenőrzik:

- DatabaseSearchErrorTest: Ez a teszteset a keresés hibakezelését vizsgálja. Létrehoz egy minimális adatbázist, majd megpróbál megkeresni benne egy nem létező filmet. A dobott hibát ellenőrzi, hogy megfelelő-e.

- DatabaseRemoveErrorTest: Ez a teszteset a törlés hibakezelését vizsgálja. Létrehoz egy minimális adatbázist, majd megpróbál törölni belőle egy nem létező filmet. A dobott hibát ellenőrzi, hogy megfelelő-e.

- DatabaseImportErrorTest: Ez a teszteset az importálás hibakezelését vizsgálja. Megpróbál egy nem létező fájlt beolvasni, majd a dobott hibát ellenőrzi, hogy megfelelő-e.