**HÁZI FELADAT**

Programozás alapjai 2.

NHF dokumentáció

Fodor Attila

EUGN1B

2023.05.26.

# 1. Feladat

- A feladat szövege InfoC++-ról származik, de saját ötletnek megfelelően van átfogalmazva
- Készítsen egy nyilvántartó rendszert, amely különböző Star Wars projekteket tart nyilván. Minden projektnek van neve/címe, kiadási éve és kiadója. Lehetséges projektek lehetnek videójátékok, filmek, sorozatok és LEGO szettek stb. Az egyes projekteknek eltérő adatokat is szükséges tárolniuk, például filmeknek a hosszát, a sorozatoknak az epizódok számát, a videójátékoknak egy rövid leírást a játékról/játékmenetről, a LEGO szetteknek pedig, hogy hány darabból állnak.
- Az objektummodellnek könnyen bővíthetőnek kell lennie, hogy a jövőben új típusú projektek is hozzáadhatóak legyenek.
- Demonstrálja a rendszer működését külön modulként fordított tesztprogrammal.
- Ne használjon STL tárolót!

# 2. Feladatspecifikáció

A program angol nyelvű lesz.
A feladat megvalósításához szükség van egy Star Wars Projects nevezetű ősosztályra és ebből az osztályból fognak leszármazni a különböző projektek[1] osztályai. A projektek egy heterogén kollekcióban lesznek tárolva.
A projekteket egy szöveges fájlból fogja beolvasni a program. Ezen felül képes lesz alapvető funkciók megvalósítására, mint az új projekt felvétele a fájlba, projekt törlése és projektek listázása. A funkciók közül a felhasználó a standard outputon megjelenített megfelelő billentyűk lenyomásával fog tudni választani.

A memóriafoglalás helyességét memtrace fogja ellenőrizni.

[1] Értsd: bármilyen, a Star Wars világával kapcsolatos film, sorozat, játék, tartalom stb.

## Adatok típusai

- Név/Cím: sztring
- Kiadási év: int
- Kiadó: sztring

Lehetséges egyéni adatok:

- Film hossza (percben): int
- Rövid játék/játékmenet leírás: sztring
- Sorozatok epizódjainak száma: int
- LEGO szettek darabszáma: int

## Hibák

A programban lehetségesen fellépő hibák esetén (hibás adat) a program a hibának megfelelő hibajelzést fog dobni.

## Tesztelés

A program működését egy külön modulként fordított főprogram fogja végezni. A program tesztelni fog a standard inputról beolvasott adatokkal. A tesztadatok közt lesz helyes és hibás adat is

VÁLTOZÁS: lásd később

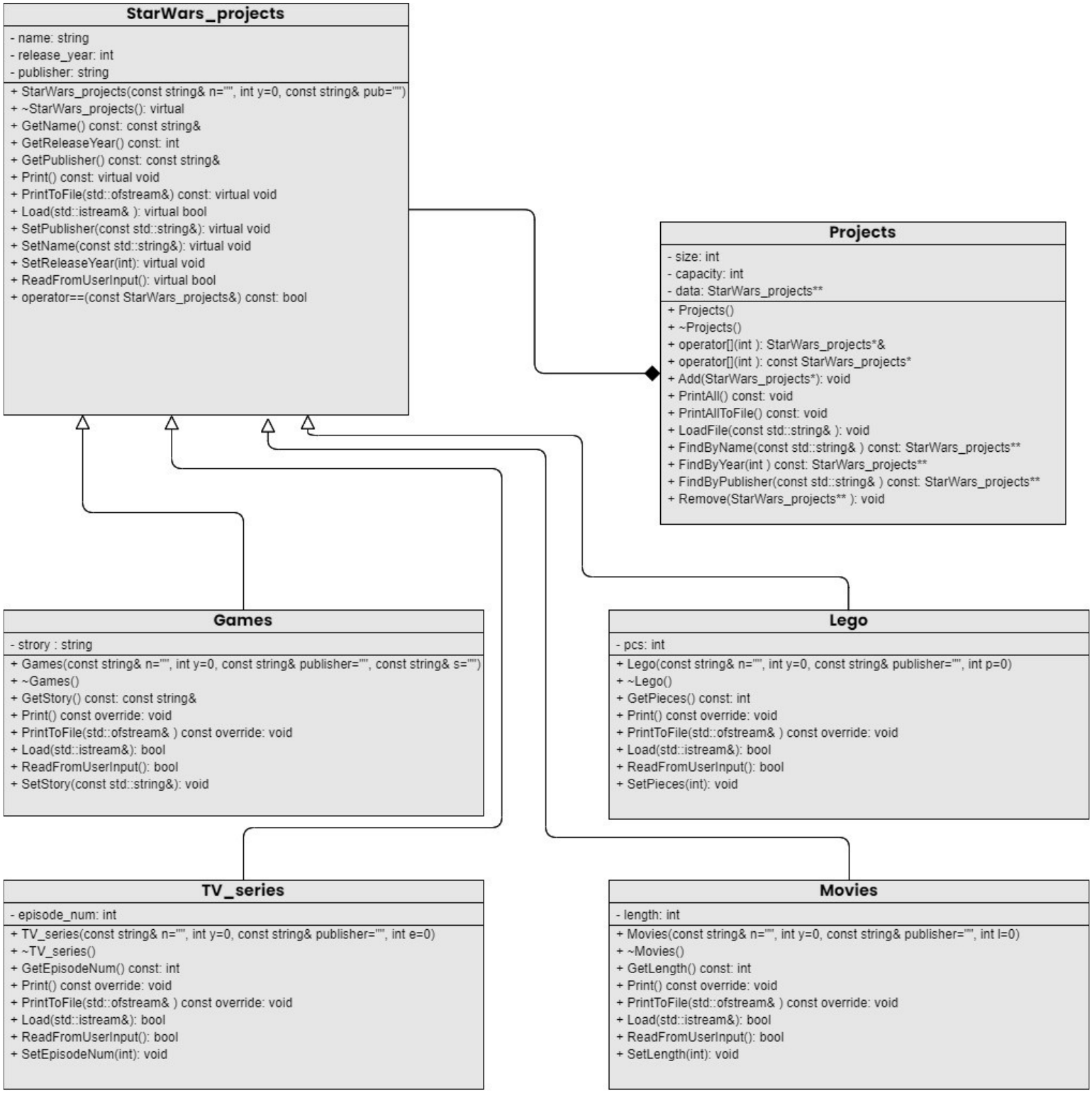
# 3. Terv

A feladat megvalósításához 6 db osztály szükséges. Ezen felül szükséges lesz egy tesztprogram elkészítése, amivel ellenőrízhető lesz a program működése

Az ősosztály (StarWars_projects) tartalmazza az adott projekt nevét (string), kiadási éve (int) és kiadója (string). Ebből az osztályból származik a 4 alosztály, melyek egy-egy különböző Star Wars projektet reprezentálnak egy-egy önálló adattal. Ezek a következők: Games: story rövid leírása(int), Lego: szett darabszáma(int), TV_series: epizódszám(int), Movies: film hossza percben(int). Az utolsó osztály egy Projects nevezetű heterogén kollekcióként szolgáló osztály, melynek adattagjai a tömb jelen mérete (=benne lévő elemek száma) (int) és egy dinamikus tömb, ami ősosztály pointereket tárol. A string típusú adatokhoz a C++ nyelv beépített string osztályát használom.

VÁLTOZÁS: A kollekciónak van egy kapacitás(int) adattagja is.

# 4. Objektum terv



# 5.Megvalósítás (legfontosabb algoritmusok)

### 5.1.1. Fájlból beolvasás és kollekció feltöltése

- A LoadFile függvény végzi, amely a Projects osztályban található
- Paraméterként átvesz egy fájlnevet, majd egyből jelzi is, ha nem tudja megnyitni azt
- Ha sikeres volt a megnyitás akkor belép egy ciklusa, ami addig fut amig tud beolvasni adatot
- Beolvassa egy adott sor első szavát, ami jelzi neki, hogy milyen típusú objektumot kell majd készíteni, ha nem ismer ilyet jelzi egy kivétellel
- Ezután meghívja a megfelelő osztály Load függvényét
- Ez a Load függvény minden alosztályban az ősosztály virtuális Load függvényének felüldefiniáltja
- Az alosztály Load függvénye meghívja az ősosztály Load függvényét, ami beolvassa a fájlból az adatokat, a setter függvényei segítségével beállítja az adattagjait majd igazzal tér vissza, ha sikeres volt
- Visszatérés után az alosztály függvénye beolvassa a saját egyéni adatát és a setter függvényével beállítja az adattagját, ha sikeres volt igazzal tér vissza a LoadFile függvénybe
- Ha az előző műveletek sikeresek voltak akkor a LoadFile függvény által dinamikusan lefoglalt megfelelő objektum betöltődik az adatokkal
- Végül az Add függvény segítségével hozzáadja a kollekcióhoz
- Ha ismeretlen típust talál akkor kiírja melyik sorba van és beolvasás nélkül megy a következő sorra, üres sor esetén is hasonlóan

### 5.1.2. Kiíratás

- A PrintAll függvény egy for ciklus segítségével végigfut a kollekción, majd minden i-edik elemre meghívja a megfelelő Print függvényt
- Ezeket az ősosztály pointereken keresztül eléri, hiszen a Print függvény virtuális az ősosztályban és ezt definiálják felül a leszármazottak

### 5.1.3. Hozzáadás a kollekcióhoz

- Paraméterként egy ősosztály pointert kap

- Felvesz egy új, az eddiginél egyel nagyobb tömböt, majd átmásolja ebbe a régi elemeit egy for ciklusban
- Végül felszabadítja a régi tömböt
- VÁLTOZÁS:
  - A függvény ellenőrzi, hogy van-e már ilyen elem a kollekcióban, ha van akkor ezt kiírja, törli ezt az objektumot és visszatér hozzáadás nélkül
  - Ha nincs ilyen elem, akkor megnézi, hogy elérte-e már a kapacitást az aktuális elemszám
  - Ha nem, akkor a következő helyre beteszi
  - Ha igen akkor felvesz egy új, az eddiginél 8-cal nagyobb kapacitású tömböt, majd átmásolja ebbe a régi elemeit egy for ciklusban és hozzáadja az új elemet is
  - Végül felszabadítja a régi tömböt

#### 5.1.4. Indexelés

- std::out_of_range kivételt dob, ha nem jó az indexelés
- konstansra is működik

#### 5.1.5. Kollekció konstruktora és destruktora

- Destruktor: egy for ciklussal végigmegy a tömbön, felszabadítja minden elemét majd végül a tömböt is
- Konstruktor: paraméter nélküli, a tömb aktuális méretét és kapacitását nullára inicializálja, a tömb pointerét NULL-ra állítja

#### 5.1.6. Osztályok konstruktorai és destruktorai

- Konstruktorok: minden osztály (kivéve Projects) konstruktora tartalmaz default értékeket, hogy paraméterrel és paraméter nélkül is hívhatóak legyenek
  - Másoló konstruktort nem kell írni, mivel csak string és int adattagok vannak az osztályokban. Ilyenkor az alapértelmezett másoló konstruktorok megfelelnek is hiszen a dinamikus adattagok (std::string típusúak) kezelése a beépített string osztály feladata
- Destruktorok: az osztályokban található destruktorok törzsei üresek (kivéve Projects osztály), mert a szöveges adatok felszabadításáról a beépített string osztály destruktora foglalkozik

#### 5.1.7. Fájlba mentés

- A Projects osztályban lévő PrintAllToFile függvény nagyon hasonlóan viselkedik, mint a PrintAll függvény:
  - Az ősosztályban található egy virtuális PrintToFile függvény, ami paraméterként egy ofstream referenciát vesz

át majd kiírja az ősosztálybeli adattagokat vesszővel elválasztva a fájlba
    - Az alosztályok felüldefiniálják ezt, kiírva a sor elejére a típust és egy pontosvesszőt és a sor végére az egyedi adatot

- Különbségek:
    - A PrintAllToFile függvény megnyitja a StarWars.txt fájlt std::ofstream::trunc módban, azaz kimeneti módban és ha nem üres törli annak tartalmát
    - Kiírja az első (nulladik) elemet a megfelelő PrintToFile meghívásával, majd belép egy for ciklusba (erre azért van szükség, hogy a beolvasásnak megfelelő legyen minden esetben a fájl formátuma)
    - A ciklusban kiír egy std::endl-t majd a második (első indexű) elemtől kezdve minden elemre meghívja a PrintToFile függvényét
    - végül bezárja a fájlt

### 5.1.8. Elem törlése

- A Remove függvény egy StarWars_projects pointert kap paraméterül
- Egy for ciklussal végigfut a kollekció elemein és ha bárhol egyezést talál akkor azt az elemet törli
    - Nincs szükség operator== felüldefiniálásra, mert az objektumokban lévő dinamikus adattagok mind std::string típusúak és a beépített string osztálynak van saját operator==-je, ezek összehasonlítását ő végzi. A többi (int) adattag összehasonlítására megfelel az alapvető operator== is
- A törölt elem helyére bemásolja az utolsó elemet majd egyel csökkenti a tömb méretét
- VÁLTOZÁS:
    - A Remove függvény egy StarWars_projects tömbre mutató pointert kap paraméterül. Megszámolja hány elem van a tömbben (while ciklussal amíg nullptr-t nem talál).
    - Ezután egy külső és belső for ciklussal végigfut a tömbön és ahol egyezés van azt az elem törlődik a kollekcióból és nullptr-re állítódik a helye
    - Végül a Remove függvény előre rendezi a kollekció nem nullptr elemeit és beállítja a megfelelő értékre a size adattagot

### 5.1.9. Keresések

- A kereső függvények (findByName, findByReleaseYear, findByPublisher) paraméterül kapnak egy string referenciát vagy int-et, végig futnak a tömbön egy for ciklussal, minden elemre meghívják a megfelelő get függvényt és ha valahol egyezés találnak visszaadják az elemre mutató pointert
- NULL-t adnak vissza, ha nem találtak megfelelő elemet
  - Pontos egyezést keresnek
- VÁLTOZÁS:
  - A függvények készítenek egy size méretű StarWars_projects pointer tömböt és minden elemét nullptr-re inicializálják. Egyezés esetén ebbe teszik bele a megfelelő pointereket, ha nincs egy darab talált elem sem akkor nullptr-re állítják a tömböt
  - Erre azért van szükség, mert lehetséges, hogy több projektnek is azonos a kiadója, a kiadási éve vagy akár a neve is
  - Azért size méretű a tömb, mert maximum size darab egyezés lehet

### 5.1.10. Objektum készítése User Inputból

- A CreateProjectFromUserInput() függvény felépítése nagyon hasonló a fájlból való beolvasásához
- Az ősosztály virtuális ReadFromUserInput() függvénye nem beolvassa, hanem bekéri a felhasználótól a megfelelő adatokat és a set függvényekkel beállítja az objektum attribútumait. Bool visszatérés jelzi a sikerességet
- Ezt definiáljak felül az alosztályok. Meghívják az ősosztály függvényét és ha az sikeres volt akkor bekérik az egyéni adatukat majd a set függvényeikkel beállítják ezeket
- Végül a CreateProjectFromUserInput() függvény megkérdezi a felhasználót milyen projektet (objektumot) akar készíteni
- Ha érvényes típust ad akkor dinamikusan foglal egy megfelelő objektumot és meghívja rá a megfelelő ReadFromUserInput() függvényt
- Ha ez sikeres akkor visszaadja az erre az objektumra mutató pointert
- Ha sikertelen akkor nullptr-t ad vissza

# 6. Osztályok és függvényeik

## 1. StarWars_projects

- Konstruktor: paraméteresen és nélküle is hívható (default értékek miatt) és Destruktor: virtuális az öröklés miatt

- **Get függvények** (GetName, GetReleaseYear, GetPublisher)
  - Visszaadják a megfelelő adattagot
- **Set függvények** (SetName, SetReleaseYear, SetPublisher)
  - Beállítják a megfelelő attribútum értékét
- **virtual void Print() const**
  - Kiírja a Star Wars projekt részleteit a kimenetre
- **virtual void PrintToFile(std::ofstream& file) const**
  - Kiírja a Star Wars projekt részleteit egy fájlba
- **virtual bool Load(std::istream& stream)**
  - Betölti a Star Wars projekt adatait az input streamből. Visszatérési érték igaz, ha a betöltés sikeres volt, hamis egyébként
- **virtual bool ReadFromUserInput()**
  - Betölti a Star Wars projekt adatait a felhasználói bemenetről. Visszatérési érték igaz, ha a betöltés sikeres volt, hamis egyébként
- **bool operator== (const StarWars_projects& other) const**
  - Összehasonlítja két objektumot, hogy azonosak-e, az összes attribútumot összehasonlítva

# 2. Games osztály

- **const std::string& GetStory() const:**
  - Visszaadja a játék történetét, leírását
- **void Print () const:**
  - Kiírja a játék részleteit és játékspecifikus adatokat a kimenetre
- **void PrintToFile(std::ofstream& file) const:**
  - Kiírja a játék adatait és a játékspecifikus adatot fájlba
- **bool Load(std::istream& stream):**
  - Betölti a játék részleteit az input streamből. Visszatérési értéke igaz, ha a betöltés sikeres volt, hamis egyébként. Beállítja a történetet (SetStory) a beolvasott értékre.
- **bool ReadFromUserInput():**
  - Beolvassa a játék részleteit és játékspecifikus adatot a felhasználói bemenetről. Visszatérési érték igaz, ha a betöltés sikeres volt, hamis egyébként. Beállítja a történetet (SetStory) a beolvasott értékre.
- **void SetStory(const std::string& s):**
  - Beállítja a játék történetét.

# 3. Lego osztály

- **int GetPieces() const:**
  - Visszaadja a Lego készlet darabszámát.
- **void Print () const:**

- Kiírja a Lego projekt részleteit és a Lego-specifikus adatot a kimenetre.

- **void PrintToFile(std::ofstream& file) const:**
  - Kiírja a Lego projekt adatait és a Lego-specifikus adatot egy fájlba.
- **bool Load(std::istream& stream):**
  - Betölti a Lego projekt részleteit az input streamből. Visszatérési értéke igaz, ha a betöltés sikeres volt, hamis egyébként. Az ősosztály adatait betölti a StarWars_projects::Load() függvénnyel. A darabszámot egy stringként olvassa be a streamből, majd egy std::stringstream segítségével konvertálja egésszé. A SetPieces() függvénnyel beállítja a készlet darabszámát.
- **bool ReadFromUserInput():**
  - Betölti a Lego projekt adatait és a Lego-specifikus adatot a felhasználói bemenetről. Visszatérési értéke igaz, ha a betöltés sikeres volt, hamis egyébként. Az ősosztály részleteit betölti a StarWars_projects::ReadFromUserInput() függvénnyel. A darabszámot bekéri a felhasználótól, majd a SetPieces() függvénnyel beállítja a készlet darabszámát.
- **void SetPieces(int pieces):**
  - Beállítja a Lego készletben lévő darabok számát.

# 4. Movies osztály

- **int GetLength () const:**
  - Visszaadja a film hosszát percben.
- **void Print () const:**
  - Kiírja a projekt (film) részleteit és a film-specifikus adatot a szabványos kimenetre.
- **void PrintToFile(std::ofstream& file) const:**
  - Kiírja a projekt (film) adatait és a film-specifikus adatot egy fájlba.
- **bool Load(std::istream& stream):**
  - Betölti a film adatait az input streamből. Visszatérési értéke igaz, ha a betöltés sikeres volt, hamis egyébként. A bázisosztály részleteit betölti a StarWars_projects::Load() függvénnyel. A film hosszát egy stringként olvassa be a streamből, majd egy std::stringstream segítségével konvertálja egésszé. A SetLength() függvénnyel beállítja a film hosszát.
- **bool ReadFromUserInput():**
  - Betölti a projekt részleteit és a film-specifikus adatot a felhasználói bemenetről. Visszatérési értéke igaz, ha a betöltés sikeres volt, egyébként hamis. Az ősosztály adatait betölti a StarWars_projects::ReadFromUserInput() függvénnyel. A film

hosszát bekéri a felhasználótól, majd a SetLength() függvénnyel beállítja azt.

- **void SetLength(int len):**
  - Beállítja a film hosszát.

# 5. TV_series osztály

- **int GetEpisodeNum() const:**
  - Visszaadja a TV sorozat epizódjainak számát.
- **void Print () const:**
  - Kiírja a projekt (TV sorozat) részleteit és a sorozat-specifikus adatot a standard kimenetre.
- **void PrintToFile(std::ofstream& file) const:**
  - Kiírja a projekt (TV sorozat) részleteit és a sorozat-specifikus adatokat egy fájlba.
- **bool Load(std::istream& stream):**
  - Betölti a TV sorozat részleteit az input streamből. Visszatérési értéke igaz, ha a betöltés sikeres, egyébként hamis. A bázisosztály adatait a StarWars_projects::Load() függvénnyel tölti be. A TV sorozat epizódjainak számát egy stringként olvassa be a streamből, majd egy std::stringstream segítségével konvertálja egésszé. A SetEpisodeNum() függvénnyel beállítja a TV sorozat epizódjainak számát.
- **bool ReadFromUserInput():**
  - Betölti a projekt részleteit és a sorozat-specifikus adatokat a felhasználói bemenetről. Visszatérési érték igaz, ha a betöltés sikeres volt, hamis egyébként. Az ősosztály adatait a StarWars_projects::ReadFromUserInput() függvénnyel tölti be. A TV sorozat epizódjainak számát bekéri a felhasználótól, majd a SetEpisodeNum() függvénnyel beállítja.
- **void SetEpisodeNum(int episodes):**
  - Beállítja a TV sorozat epizódjainak számát.

# 6. Projects

- **Projects():**
  - Az osztály konstruktora. Létrehoz egy üres Projects objektumot.
- **~Projects():**
  - Az osztály destruktora. Törli a dinamikusan foglalt StarWars_projects objektumokat és felszabadítja a memóriát.
- **StarWars_projects*& operator[] (int index):**
  - Túlterhelt operátor [], amely lehetővé teszi a StarWars_projects objektumok hozzáférését és módosítását a megadott index alapján. Kivételt dob, ha az index kívül esik a megengedett tartományon.

- **const StarWars_projects* operator[](int index) const:**
  - Konstans változat a [] operátor túlterhelésének, amely lehetővé teszi a StarWars_projects objektumok hozzáférését a megadott index alapján. Kivételt dob, ha az index kívül esik a megengedett tartományon.
- **StarWars_projects* CreateProjectFromUserInput():**
  - Új StarWars_projects objektumot hoz létre a felhasználói bemeneten kapott adatok alapján. A felhasználótól bekér egy projekttípust, majd létrehoz egy új objektumot az adott típus alapján. Ezután visszatér a létrehozott objektumra mutató pointerrel.
- **void Add (StarWars_projects* project):**
  - Új StarWars_projects objektumra mutató pointer hozzáadása a Projects adatbázishoz. Ellenőrzi, hogy a projekt már szerepel-e az adatbázisban (név, kiadási év és kiadója alapján). Ha igen, hibaüzenetet jelenít meg és törli a projektet. Ha nem, hozzáadja a projektet az adatbázishoz.
- **void PrintAll () const:**
  - Kiírja az összes kollekcióban található StarWars_projects objektumot a kimenetre.
- **void PrintAllToFile() const:**
  - Kiírja az összes StarWars_projects objektum adatát egy 'StarWars.txt' nevű fájlba. Ha a fájl nem üres, akkor az eddigi tartalom törlődik. Hibaüzenetet jelenít meg, ha nem sikerül megnyitni a fájlt.
- **void LoadFile (const std::string& filename):**
  - Betölti a StarWars_projects adatokat egy fájlból és feltölti velük a Projects adatbázist. Hibaüzenetet jelenít meg, ha nem sikerül megnyitni a fájlt vagy ismeretlen projekttípust talál a fájlban. Üres sor esetén a következővel folytatja.
- **StarWars_projects** FindByName(const std::string& name) const:**
  - Keresés projektnév alapján. Létrehoz egy 'size' méretű StarWars_projects objektumokra mutató pointerek tömbjét. Az array összes elemét inicializálja nullptr értékre. Nyomon követi a talált projektek számát, majd végig iterál minden projekten az adat tömbben. Ellenőrzi, hogy a projekt neve megegyezik-e a megadott névvel. Ha találat van, hozzáadja a projektet a foundProjects tömbhöz. Ha nem talál projekteket, felszabadítja a memóriát és nullptr-re állítja a pointert. Végül visszatér a talált projektek tömbjével (vagy nullptr-rel, ha nem talál egyezést).

- **StarWars_projects** FindBYear(int year) const:**
  - Keresés kiadási év alapján. Létrehoz egy 'size' méretű StarWars_projects objektumokra mutató pointerek tömbjét. A tömb összes elemét inicializálja nullptr értékre. Nyomon követi a talált projektek számát. Végig iterál minden projekten az adat tömbben. Ellenőrzi, hogy a projekt neve megegyezik-e a megadott névvel. Ha találat van, hozzáadja a projektet a foundProjects tömbhöz. Ha nem talál projekteket, felszabadítja a memóriát és nullptr-re állítja a pointert. Végül visszatér a talált projektek tömbjével (vagy nullptr-rel, ha nem talál egyezést).
- **StarWars_projects** FindByPublisher(const std::string& publisher) const:**
  - Keresés projektnév alapján. Létrehoz egy 'size' méretű StarWars_projects objektumokra mutató pointerek tömbjét. Az array összes elemét inicializálja nullptr értékre. Nyomon követi a talált projektek számát, majd végigiterál minden projekten az adat tömbben. Ellenőrzi, hogy a projekt neve megegyezik-e a megadott névvel. Ha találat van, hozzáadja a projektet a foundProjects tömbhöz. Ha nem talál projekteket, felszabadítja a memóriát és nullptr-ra állítja a pointert. Végül visszatér a talált projektek tömbjével (vagy nullptr-rel, ha nem talál egyezést).
- **void Remove(StarWars_projects** projectsToRemove):**
  - Megszámolja a törlendő projektek számát a kapott tömbben (számol amíg nullptr-t nem talál). Az első for ciklusban végig iterál a törlendő projektek listáján. A második for ciklusban végig iterál a kollekcióban lévő projekteken. Ha egy projekt a kollekcióban megegyezik az i-edik törlendő projekttel, akkor törli a projekt objektumot és nullptr-re állítja a törölt objektum helyét a kollekcióban. Ezután újra végig iterál az adat tömbön annak érdekében, hogy a nullptr elemeket eltávolítsa és előre rendezze a tömböt. Ha nem-nullptr projektet talál, áthelyezi a kollekcióban a következő rendelkezésre álló indexre. Frissíti a kollekció size adattagját, hogy az a benne lévő projektek számát tükrözze.

# 7. Fájl

A fájl, amely az adatokat tartalmazza .txt formátumú, sorai a következőképp épülnek fel: TÍPUS;név,kiadási év,kiadó,'egyedi adat'

A típus megmondja milyen fajtájú projektről van szó, ez teljesen nagybetűs utána pontosvessző.A többi adatot vessző választja el egymástól. Az egyedi adat az

alosztályokra vonatkozó, az ősosztály által nem tartalmazott plusz adat. A sor végén sortörés van.

# 8. Tesztelés

2 tesztesetet készítettem. Ezek között a TEST makró 1 vagy 2 választással lehet váltani. 1 esetén előre megadott adatokkal dolgozik a program, 2 esetén a felhasználótól vár el adatokat.

**TEST 1**

Ebben a tesztesetben a projektek hozzáadását, keresését és eltávolítását teszteltük. Először létrehoztunk egy üres heterogén kollekciót. Betöltöttük a fájl („StarWars.txt”) tartalmát és kiírtuk. Ezek után hozzáadtunk néhány StarWars_projects objektumot a Projects adatbázishoz. Az egyik egy film ami már benne van, a másik egy játék ami újonnan kerül a kollekcióba. Ellenőriztük, hogy ez sikeres volt azzal, hogy rákerestünk a hozzáadott projektek évére. Teszteltük a kiadó keresést is („Lego” kiadóval). Majd töröltünk az újonnan hozzáadott játék projektet az adatbázisból (név szerinti keresés után törlés). Majd kiírtuk a módosult állományt és visszatöltöttük fájlba.

**TEST 2**

Ez a teszteset felhasználói tesztre készült. A program enélkül is helyesen fut és az előbbi teszteset megmozgatja a program minden szegmensét. Itt is először betöltődik a fájl és kiíratásra kerül a tartalma. Ezután a felhasználót kérdezzük akar-e hozzáadni, keresni és- vagy törölni és ha igen mi alapján. Addig fut a program amíg a felhasználó úgy nem dönt, hogy már egyiket se akarja csinálni. Ezután kiírjuk neki a módosult állományt és elmentjük fájlba az adatokat.

Mindkét tesztesetben ellenőriztük az elvárt eredményeket, és bizonyítottuk, hogy az osztályok működése a tervezett módon történik. A tesztek segítettek megbizonyosodni arról, hogy a Projects osztály helyesen kezeli a projektek hozzáadását, eltávolítását és keresését, valamint megfelelően kezeli a memóriát is. A programban nincs memóriaszivárgás, ez a memtrace segítségével van ellenőrizve.