Filmtár

Göbhardter Ádám
2020.05.15

Hierarchikus mutató

Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

Osztálymutató

Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

Fájlmutató

Fájllista

Az összes dokumentált fájl listája rövid leírásokkal:

Osztályok dokumentációja

Documentary osztályreferencia

Documetary származtatott osztály.

Publikus tagfüggvények

- Documentary (string m="", string c="", unsigned int h=0, unsigned int k=0, string l="")
- string get_leiras () const
- void show ()

Függvény, amely megjeleníti a Documentary típusú filmek alapadatait.

- void print_in_file (ofstream &f)

Részletes leírás

Documetary származtatott osztály.

(Definíció a(z) Documentary.hpp fájlban.)

Konstruktorok és destruktorok dokumentációja

Documentary:: Documentary (string *m* = "", string *c* = "", unsigned int *h* = 0, unsigned int *k* = 0, string *l* = "") `[inline]`, `[explicit]`

Alapértelmezett és paraméteres konstruktor.

Paraméterek:

m	Műfaj megnevezése, alapértelmezett érték: üres sztring.
c	Cím megnevezése, alapértelmezett érték: üres sztring.
h	Hossz megadása, alapértelmezett érték: 0.
k	Kiadás évének megadása. Alapértelmezett érték: 0.
l	Leírás megadása. Alapértelmezett érték: üres sztring.

(Definíció a(z) Documentary.hpp fájlban.)

Tagfüggvények dokumentációja

string Documentary::get_leiras () const `[inline]`

Leírás lekérdezése.

Visszatérési érték: Leírás.

(Definíció a(z) Documentary.hpp fájlban.)

void Documentary::print_in_file (ofstream & *f*) `[inline]`, `[virtual]`

Függvény, amely kiírja a Documentary típusú filmek alapadatait a megadott stream-re.

Paraméterek:

f	Stream, ahová az adatok kiírásra kerülnek (output stream referencia).

Újraimplementált ősök: Movie.

(Definíció a(z) Documentary.hpp fájlban.)

FamilyMovie osztályreferencia

FamilyMovie származtatott osztály.

Publikus tagfüggvények

- FamilyMovie (string m="", string c="", unsigned int h=0, unsigned int k=0, unsigned int kor=0)
- unsigned int get_korhatar () const
- void show ()

Függvény, amely megjeleníti a FamilyMovie típusú filmek alapadatait.

- void print_in_file (ofstream &f)

Részletes leírás

FamilyMovie származtatott osztály.

(Definíció a(z) FamilyMovie.hpp fájlban.)

Konstruktorok és destruktorok dokumentációja

FamilyMovie:: FamilyMovie (string *m* = "", string *c* = "", unsigned int *h* = 0, unsigned int *k* = 0, unsigned int *kor* = 0)[inline], [explicit]

Alapértelmezett és paraméteres konstruktor.

Paraméterek:

m	Műfaj megnevezése, alapértelmezett érték: üres sztring.
c	Cím megnevezése, alapértelmezett érték: üres sztring.
h	Hossz megadása, alapértelmezett érték: 0.
k	Kiadás évének megadása. Alapértelmezett érték: 0.
kor	Korhatár megadása. Alapértelmezett érték: 0.

(Definíció a(z) FamilyMovie.hpp fájlban.)

Tagfüggvények dokumentációja

unsigned int FamilyMovie::get_korhatar () const[inline]

Korhatár lekérdezése.

Visszatérési érték: Korhatár.

(Definíció a(z) FamilyMovie.hpp fájl . sorában.)

void FamilyMovie::print_in_file (ofstream & *f*)[inline], [virtual]

Függvény, amely kiírja a FamilyMovie típusú filmek alapadatait a megadott stream-re.

Paraméterek:

f	Stream, ahová az adatok kiírásra kerülnek (output stream referencia).

Újraimplementált ősök: Movie.

(Definíció a(z) FamilyMovie.hpp fájlban.)

Lista osztályreferencia

A láncolt listát megvalósító osztály.

Publikus tagfüggvények

- Lista ()
  Alapértelmezett konstruktor, amely egy üres listát hoz létre.
- ListaElem * get_eleje () const
- void add (string g, string t, unsigned int l, unsigned int d, unsigned int a, string c)
  Új elem hozzáadása a láncolt listához.
- void felszab ()
  A láncolt lista felszabadítása.
- void print ()
  A láncolt lista elemek adatainak kiíratása.
- void remove (string m)
  Egy megadott című film kitörlése a láncolt listából.
- void search (string o)
  Egy megadott című film megkeresése a láncolt listában, majd adatainak kiíratása.
- void filter (string h)
  A láncolt listában megtalálható, megadott mûfajú filmek címének kilistázása.
- void save (string filename)
  A láncolt lista elemek adatainak kiíratása fájlba.
- void read (string filename)
  Korábban kimentett láncolt lista elemek visszaolvasása láncolt listába.

(Definíció a(z) Lista.hpp fájl . sorában.)

Tagfüggvények dokumentációja

void Lista::add (string *g*, string *t*, unsigned int *l*, unsigned int *d*, unsigned int *a*, string *c*)

Új elem hozzáadása a láncolt listához.

Paraméterek:

g	Műfaj megnevezése.
t	Cím megnevezése.
l	Hossz megadása.
d	Kiadás évének megadása.
a	Korhatár megadása. Alapértelmezett érték 0.
c	Leírás megadása.

Ennek a tagfüggvénynek a segítségével tudunk új elemeket hozzáadni a láncolt listához. Kezdetben 3 különböző műfajú filmet különböztetünk meg: családi filmet, dokumentum filmet és egyéb kategóriába tartozó filmet. Paraméterként átadjuk a függvénynek azokat az adatokat, amelyekkel létre tudunk hozni egy adott műfajú filmet. Lehetséges, hogy a megadott 6 paraméter közül csak 4-et használunk (az egyéb műfajú filmek esetén), ilyenkor a többi paraméter dummy paraméter lesz. A függvény kezdetben létrehoz egy új ListaElemre mutató pointert. Ezután el kell döntenünk, hogy az adott elem milyen filmet fog tárolni. Ehhez a megadott "g" paraméter segítségével meghatározzuk a film műfaját, majd ez alapján hozunk létre egy új film objektumot (dinamikusan), amit rögtön a ListaElemhez adunk. Mivel egyszerre több film hozzáadása nem lehetséges, ezért a következő ListaElemre mutató pointert NULL-ra állítjuk. Ezután, ha az eredeti lista elejére mutató pointer (eleje) NULL, akkor egyszerűen csak egyenlővé tesszük vele az "uj" pointert, hiszen ilyenkor ezek megegyeznek. Egyéb esetben az eredeti lista végéhez adjuk az új létrehozott elemet.

(Definíció a(z) Lista.cpp fájlban.)

void Lista::felszab ()

A láncolt lista felszabadítása.

Ennek a tagfüggvénynek a segítségével tudjuk a lista dinamikusan foglalt részeit felszabadítani. Egyszerűen csak végiglépkedünk a listán, majd felszabadítjuk a dinamikusan foglalt, ListaElem-ekben tárolt, film objektumokat, majd a szintén dinamikusan foglalt ListaElemeket is. A felszabadítás végeztével a lista elejére mutató pointert NULL-ra állítjuk, ezzel jelezve, hogy a lista üres.

(Definíció a(z) Lista.cpp fájlban.)

void Lista::filter (string *h*)

A láncolt listában megtalálható, megadott mûfajú filmek címének kilistázása.

Paraméterek:

h	Műfaj megadása.

Ennek a tagfüggvénynek a segítségével ki tudjuk íratni a képernyõre a paraméterként megadott műfajú filmek címét. Először végiglépkedünk a listán, és, ha az aktuális ListaElem által tartalmazott film mûfaja megegyezik a paraméterkén kapott műfajjal, akkor kiíratjuk az adott film címét. Az iteráció során figyeljük, hogy volt-e min. 1 egyezés; ha nem volt, akkor erről tájékoztatjuk a felhasználót.

(Definíció a(z) Lista.cpp fájlban.)

ListaElem* Lista::get_eleje () const`[inline]`

A láncolt lista elejének lekérdezése.

Visszatérési érték: A láncolt lista elejére mutató pointer.

(Definíció a(z) Lista.hpp fájlban)

void Lista::print ()

A láncolt lista elemek adatainak kiíratása.

Ennek a tagfüggvénynek a segítségével ki tudjuk íratni a lista tartalmát a képernyõre. Ha a lista elejére mutató pointer NULL, akkor a lista üres. Egyéb esetben végiglépkedünk a listán, és meghívjuk az elemek által tárolt film objektumok show() tagfüggvényét, amely a tényleges kiíratást végzi.

(Definíció a(z) Lista.cpp fájlban.)

void Lista::read (string *filename*)

Korábban kimentett láncolt lista elemek visszaolvasása láncolt listába.

Paraméterek:

filename	A fájl neve, ahonnan az adatokat be szeretnénk olvasni.

Ennek a tagfüggvénynek a segítségével egy létező fájlból tudunk korábban kimentett adatokat visszaolvasni egy listába. A függvény a fájl megnyitását követően ellenőrzi, hogy a művelet sikeres volt-e. Ha nem, akkor erről értesíti a felhasználót, majd befejezi a program végrehajtását. Sikeres megnyitást követően a függvény ellenőrzi a megnyitott fájl tartalmát. Ha az üres, akkor nem tesz semmit, egyből visszatér. Egyéb esetben a kurzort a fájl elejére mozgatjuk (hiszen a beolvasás kurzortól indul, és kiíratás után a kurzor a fájl végén marad). Ezután megkezdődik a beolvasás, ami egészen a fájl végéig (EOF) tart. Minden egyes beolvasott filmet egy új ListaElemben tárolunk el. A film típusát az első beolvasott adattal (műfaj) döntjük el, majd ez alapján adjuk hozzá a ListaElemhez dinamikus foglalással. Ezt követően a függvény az eredeti listához adja a kész listaelemet. Ha a lista üres, akkor az elejéhez adja, egyéb esetben pedig a végéhez fûzi az elemet. Ha a beolvasás befejeződött, a függvény automatikusan visszatér. Fontos, hogy kiíratás során, az utolsó kiírt film után keletkezik egy üres sor. Annak érdekében, hogy ezt ne olvassuk be, minden új sor beolvasása előtt leellenőrizzük, hogy az első adat a sorban üres sztring-e. Ha igen, akkor az adott sort kihagyjuk (break), és befejezzük a beolvasást.

Figyelmeztetés:

A film_database.txt fájl külső, hibás módosítása során (pl. egy családi filmnek nem írunk be korhatárt) hibák jelentkezhetnek.

(Definíció a(z) Lista.cpp fájlban.)

void Lista::remove (string *m*)

Egy megadott című film kitörlése a láncolt listából.

Paraméterek:

m	Cím megadása.

Ennek a tagfüggvénynek a segítségével egy, paraméterként megadott című filmet távolíthatunk el a listából. Ha a lista üres, a függvény egyből visszatér, hiszen ilyenkor nincs mit kitörölni. Ha a lista legelső eleme az, amit ki kell törölni, akkor a 2. ListaElemre mutató pointert elmentjük, majd felszabadítjuk az első elemet és az általa tárolt film objektumot is. Ezután a lista elejére mutató pointert egyenlővé tesszük az elmentett pointerrel (ami most az első elem lett). Egyéb esetben végiglépkedünk a listán, majd, ha az adott elem által tárolt film objektum címe megegyezik a a paraméterként megadottal, akkor felszabadítjuk mind a film objektumot, mind az azt tároló ListaElemet. Ebben az esetben az első két elem ponterét kell "megjegyeznünk". A "temp" pointer fog majd a törlendő elemre mutatni. ha megtaláltuk az elemet, akkor összekapcsoljuk a törlendő ListaElem előtti és utáni elemeket, majd elvégezzük a törlést. Ha a listában nem szerepel a törölni kívánt film, akkor a függvény nem tesz semmit (visszatér). A függvény értesíti a felhasználót, ha az adatbázis üres, a keresett című film nem található, illetve, ha a törlés sikeresen megtörtént.

(Definíció a(z) Lista.cpp fájlban.)

void Lista::save (string *filename*)

A láncolt lista elemek adatainak kiíratása fájlba.

Paraméterek:

filename	A fájl neve, ahová az adatokat menteni szeretnénk.

Ennek a tagfüggvénynek a segítségével a lista tartalmát tudjuk kiírni egy fájlba. A függvény egy tetszőleges nevű fájlt nyit meg (vagy hoz létre, ha korábban nem létezett), és tartalmát egyből el is dobja (nincs már rá szükség, hiszen a program elindítása után a read() függvény már beolvasta a fájl tartalmát). Ha a fájl megnyitása során hiba lépett fel, azt jelezzük a felhasználónak, majd a program végrehajtása befejeződik. Egyéb esetben végiglépkedünk a listán, és meghívjuk az összes elem által tartalmazott film objektum print_in_file() tagfüggvényét, amely a tényleges fájlba való kiíratást végzi.

(Definíció a(z) Lista.cpp fájlban.)

void Lista::search (string *o*)

Egy megadott című film megkeresése a láncolt listában, majd adatainak kiíratása.

Paraméterek:

o	Cím megadása.

Ennek a tagfüggvénynek a segítségével ki tudjuk íratni a paraméterként megadott címû film(ek) adatait a képernyőre. Először végiglépkedünk a listán, és, ha az aktuális ListaElem által tartalmazott film címe megegyezik a paraméterkén kapott címmel, akkor meghívjuk a film objektum show() tagfüggvényét, amely a tényleges kiíratást végzi. Az iteráció során figyeljük, hogy volt-e egyezés; ha nem volt, akkor erről tájékoztatjuk a felhasználót.

(Definíció a(z) Lista.cpp fájlban.)

ListaElem osztályreferencia

Osztály, amely a láncolt listának egy-egy elemét képviseli.

Publikus attribútumok

- Movie * **fp**
- ListaElem * **kov**

(Definíció a(z) Lista.hpp fájlban.)

Movie osztályreferencia

Movie alaposztály.

Publikus tagfüggvények

- Movie (string m="", string c="", unsigned int h=0, unsigned int k=0)
- virtual ~Movie ()

  Virtuális destruktor az öröklés miatt.
- string get_cim () const
- string get_mufaj () const
- unsigned int get_hossz () const
- unsigned int get_kiadas () const
- virtual void show ()

  Virtuális függvény, amely megjeleníti a Movie típusú filmek alapadatait.
- virtual void print_in_file (ofstream &f)

(Definíció a(z) Movie.hpp fájlban.)

Konstruktorok és destruktorok dokumentációja

Movie:: Movie (string *m* = "", string *c* = "", unsigned int *h* = 0, unsigned int *k* = 0)`[inline]`, `[explicit]`

Alapértelmezett és paraméteres konstruktor.

Paraméterek:

m	Műfaj megnevezése, alapértelmezett érték: üres sztring.
c	Cím megnevezése, alapértelmezett érték: üres sztring.
h	Hossz megadása, alapértelmezett érték: 0.
k	Kiadás évének megadása. Alapértelmezett érték: 0.

(Definíció a(z) Movie.hpp fájlban.)

Tagfüggvények dokumentációja

string Movie::get_cim () const`[inline]`

Cím lekérdezése.

Visszatérési érték: Cím.

(Definíció a(z) Movie.hpp fájlban.)

unsigned int Movie::get_hossz () const`[inline]`

Hossz lekérdezése.

Visszatérési érték: Hossz.

(Definíció a(z) Movie.hpp fájlban.)

unsigned int Movie::get_kiadas () const`[inline]`

Kiadás lekérdezése.

Visszatérési érték: Kiadás.

(Definíció a(z) Movie.hpp fájlban.)

string Movie::get_mufaj () const`[inline]`

Műfaj lekérdezése.

Visszatérési érték: Műfaj.

(Definíció a(z) Movie.hpp fájlban.)

virtual void Movie::print_in_file (ofstream & *f*)`[inline]`, `[virtual]`

Virtuális függvény, amely kiírja a Movie típusú filmek alapadatait a megadott stream-re.

Paraméterek:

f	Stream, ahová az adatok kiírásra kerülnek (output stream referencia).

Újraimplementáló leszármazottak: Documentary és FamilyMovie.

(Definíció a(z) Movie.hpp fájlban.)

Fájlok dokumentációja

files/Documentary.hpp fájlreferencia

Osztályok

- class Documentary - *Documetary származtatott osztály.*

Részletes leírás

Ez a fájl tartalmazza a Documentary származtatott osztály deklarációját, illetve az osztály tagfüggvényeinek definícióját is.

(Definíció a(z) Documentary.hpp fájlban.)

Documentary.hpp

```
    1 #ifndef DOCUMENTARY_HPP
    2 #define DOCUMENTARY_HPP
    3
   10 #include "Movie.hpp"
   11
   13 class Documentary : public Movie{
   14
   15     string leiras;
   16
   17     public:
   18
   25         explicit Documentary(string m = "", string c = "", unsigned int h = 0, unsigned int k = 0,
string l = "") : Movie(m, c, h, k), leiras(l) {}
   26
   27
   30         string get_leiras()const{return leiras;}
   31
   33         void show(){
   34             Movie::show();
   35             std::cout << " " << leiras;
   36         }
   37
```

```
   40         void print_in_file(ofstream& f){
   41             f << get_mufaj() << "\t" << get_cim() << "\t" << get_hossz() << "\t" << get_kiadas() << "\
t" << leiras << "\n";
   42         }
   43 };
   44
   45 #endif // DOCUMENTARY_HPP
```

---

files/FamilyMovie.hpp fájlreferencia

Osztályok

- class FamilyMovie - *FamilyMovie származtatott osztály.*

Részletes leírás

Ez a fájl tartalmazza a FamilyMovie származtatott osztály deklarációját és az osztály tagfüggvényeinek definícióját is.

(Definíció a(z) FamilyMovie.hpp fájlban.)

FamilyMovie.hpp

```
    1 #ifndef FAMILYMOVIE_HPP
    2 #define FAMILYMOVIE_HPP
    3
   10 #include "Movie.hpp"
   11
   13 class FamilyMovie : public Movie{
   14
   15     unsigned int korhatar;
   16
   17     public:
   18
   25         explicit FamilyMovie(string m = "",  string c = "", unsigned int h = 0, unsigned int k = 0,
unsigned int kor = 0) : Movie(m, c, h, k), korhatar(kor) {}
   26
   29         unsigned int get_korhatar()const{return korhatar;}
   30
   32         void show(){
   33             Movie::show();
   34             std::cout << " " << korhatar;
   35         }
   36
   39         void print_in_file(ofstream& f){
   40             f << get_mufaj() << "\t" << get_cim() << "\t" << get_hossz() << "\t" << get_kiadas() << "\
t" << korhatar << "\n";
   41         }
   42 };
   43
   44 #endif // FAMILYMOVIE_HPP
```

---

files/Lista.cpp fájlreferencia

Részletes leírás

Ez a fájl tartalmazza a Lista osztály tagfüggvényeinek definícióját.

(Definíció a(z) Lista.cpp fájlban.)

Lista.cpp

```
    1
    7 #include <iostream>
    8 #include <fstream>
    9 #include <exception>
   10
   11 #include "Lista.hpp"
```

```
12 #include "Movie.hpp"
13 #include "FamilyMovie.hpp"
14 #include "Documentary.hpp"
15
16 using std::cout;
17 using std::cerr;
18 using std::endl;
19 using std::stoi;
20 using std::ofstream;
21 using std::ifstream;
22 using std::ios_base;
23
34 void Lista::add(string g, string t, unsigned int l, unsigned int d, unsigned int a, string c){
35
36     ListaElem* uj = new ListaElem();
37
38     if(g == "csaladi"){
39         uj->fp = new FamilyMovie(g, t, l, d, a);
40         uj->kov = nullptr;
41     }
42     else{
43         if(g == "dokumentum"){
44             uj->fp = new Documentary(g, t, l, d, c);
45             uj->kov = nullptr;
46         }
47         else{
48             uj->fp = new Movie(g, t, l, d);
49             uj->kov = nullptr;
50         }
51     }
52
53     if(eleje == nullptr){eleje = uj;}
54     else{
55         ListaElem* temp = eleje;
56         while(temp->kov != nullptr){temp = temp->kov;}
57         temp->kov = uj;
58     }
59 }
60
66 void Lista::felszab(){
67     ListaElem* temp = eleje;
68
69     while(temp != nullptr){
70         ListaElem* kov = temp->kov;
71         delete temp->fp;
72         delete temp;
73         temp = kov;
74     }
75
76     eleje = nullptr;
77 }
78
84 void Lista::print(){
85
86     if(eleje == nullptr){cout << "Az adatbazis ures.\n" << endl;}
87
88     ListaElem* temp;
89
90     unsigned int sorszam = 1;
91
92     for(temp = eleje; temp; temp = temp->kov){
93         cout << sorszam++ << ". ";
94         temp->fp->show();
95         cout << endl;
96     }
97
98     cout << endl;
99 }
100
110 void Lista::remove(string m){
111
112     if(eleje == nullptr){
113         cout << "Az adatbazis ures." << endl;
114         return;
115     }
116
117     if(eleje->fp->get_cim() == m){
118
119         ListaElem* temp = eleje->kov;
120         delete eleje->fp;
121         delete eleje;
```

```
122
123         eleje = temp;
124
125         cout << "Sikeres torles." << endl;
126
127         return;
128     }
129
130     ListaElem* temp = nullptr;
131     ListaElem* elozo;
132
133     for(temp = eleje->kov, elozo = eleje; temp; elozo = temp, temp = temp->kov){
134         if(temp->fp->get_cim() == m){
135
136             elozo->kov = temp->kov;
137             delete temp->fp;
138             delete temp;
139
140             cout << "Sikeres torles." << endl;
141
142             return;
143         }
144     }
145
146     cout << "Az adatbazisban nem talalhato ilyen cimu film." << endl;
147
148     return;
149 }
150
157 void Lista::search(string o){
158
159     bool van = false;
160     unsigned int sorszam = 1;
161
162     ListaElem* temp;
163     for(temp = eleje; temp; temp = temp->kov){
164         if(temp->fp->get_cim() == o){
165
166             cout << sorszam++ << ". ";
167             temp->fp->show();
168             cout << endl;
169
170             van = true;
171         }
172     }
173
174     if(!van){cout << "Nincs ilyen cimu film az adatbazisban.\n" << endl;}
175 }
176
182 void Lista::filter(string h){
183
184     bool van = false;
185     unsigned int sorszam = 1;
186
187     ListaElem* temp;
188     for(temp = eleje; temp; temp = temp->kov){
189         if(temp->fp->get_mufaj() == h){
190             cout << sorszam++ << ". ";
191             cout << temp->fp->get_cim() << "\n" << endl;
192             van = true;
193         }
194     }
195
196     if(!van){cout << "Nincs ilyen mufaju film az adatbazisban.\n" << endl;}
197 }
198
205 void Lista::save(string filename){
206
207     ofstream f(filename, ios_base::out | ios_base::trunc);
208
209     if(!f.good()){
210         felszab();
211         throw "Hiba tortent az adatbazis mentese soran. A program kilep.";
212     }
213
214     ListaElem* temp;
215     for(temp = eleje; temp; temp = temp->kov){
216         temp->fp->print_in_file(f);
217     }
218
219     f.flush();
```

```
220     f.close();
221 }
222
235 void Lista::read(string filename){
236
237     ifstream f(filename, ios_base::in);
238
239     if(!f.good()){
240         throw "Hiba tortent az adatbazis beolvasasa soran. A program kilep.";
241     }
242
243     f.seekg(0, f.end);
244     size_t size = f.tellg();
245     if(size == 0){
246         f.close();
247         return;
248     }
249
250     string mufaj, cim, hossz, kiadas, leiras, korhatar;
251     f.seekg(0, f.beg);
252
253     while(!f.eof()){
254
255         ListaElem* uj = new ListaElem();
256
257         getline(f, mufaj, '\t');
258
259         if(mufaj == ""){
260             delete uj;
261             break;
262         }
263
264         getline(f, cim, '\t');
265         getline(f, hossz, '\t');
266
267         if(mufaj == "csaladi"){
268             getline(f, kiadas, '\t');
269             getline(f, korhatar);
270
271             uj->fp = new FamilyMovie(mufaj, cim, stoi(hossz), stoi(kiadas),stoi(korhatar));
272             uj->kov = nullptr;
273         }
274         else{
275             if(mufaj == "dokumentum"){
276                 getline(f, kiadas, '\t');
277                 getline(f, leiras);
278
279                 uj->fp = new Documentary(mufaj, cim, stoi(hossz), stoi(kiadas), leiras);
280                 uj->kov = nullptr;
281             }
282             else{
283                 getline(f, kiadas);
284
285                 uj->fp = new Movie(mufaj, cim, stoi(hossz), stoi(kiadas));
286                 uj->kov = nullptr;
287             }
288         }
289
290         if(eleje == nullptr){
291             eleje = uj;
292         }
293         else{
294             ListaElem* mozgo = eleje;
295             while(mozgo->kov != nullptr){
296                 mozgo = mozgo->kov;
297             }
298             mozgo->kov = uj;
299         }
300     }
301
302     f.close();
303 }
```

files/Lista.hpp fájlreferencia

Osztályok

- class ListaElem - *Osztály, amely a láncolt listának egy-egy elemét képviseli.*
- class Lista - *A láncolt listát megvalósító osztály.*

Részletes leírás

Ez a fájl tartalmazza a Lista és ListaElem osztályok deklarációját, illetve a Lista osztály tagfüggvényeinek deklarációját.

(Definíció a(z) Lista.hpp fájlban.)

Lista.hpp

```
 1 #ifndef LISTA_HPP
 2 #define LISTA_HPP
 3
10 #include "memtrace.h"
11
12 #include "Movie.hpp"
13 #include "FamilyMovie.hpp"
14 #include "Documentary.hpp"
15
17 class ListaElem{
18
19     public:
20
21         Movie* fp;
22         ListaElem* kov;
23 };
24
26 class Lista{
27     ListaElem* eleje;
28
29     public:
30
32         Lista() : eleje(nullptr) {}
33
36         ListaElem* get_eleje()const{return eleje;}
37
45         void add(string g, string t, unsigned int l, unsigned int d, unsigned int a, string c);
46
48         void felszab();
49
51         void print();
52
55         void remove(string m);
56
59         void search(string o);
60
63         void filter(string h);
64
67         void save(string filename);
68
71         void read(string filename);
72 };
73
74 #endif //LISTA_HPP
```

---

files/main.cpp fájlreferencia

Függvények

- void options () -*A függvény megjeleníti a képernyõn a felhasználó által választható opciókat.*
- unsigned int choice (Lista l) - *Ez a függvény fogadja a felhasználó által választott menüpont sorszámát.*
- int main () - *A teljes adatbázis vezérlését végzi a korábban megírt függvények felhasználásával.*

Részletes leírás

Ez a fájl tartalmazza az adatbázis működését vezérlő main() függvény és egyéb járulékos függvények definícióját/deklarációját, illetve itt hajtódnak végre a tesztesetek is.

(Definíció a(z) main.cpp fájlban.)

Függvények dokumentációja

unsigned int choice (Lista *l*)

Ez a függvény fogadja a felhasználó által választott menüpont sorszámát.

Paraméterek:

l	Az eredeti lista, amivel a program futása közben dolgozunk (felszabadítás miatt kell érvénytelen input esetén).

Visszatérési érték: A felhasználó döntése.

A függvény felszólítja a felhasználót, hogy válasszon a menüből. Ha a felhasználó választott, akkor visszatér a választással. Amennyiben nem létezik a választott menüpont, akkor a felhasználónak lehetősége van újra választani. A függvény ellenőrzi az érvénytelen inputokat (pl. ha szám helyett karaktert adunk meg), és, ha ilyet kap, akkor erről értesíti a felhasználót, és befejezi a program végrehajtását.

(Definíció a(z) main.cpp fájlban.)

main.cpp

```
 1
 7 #include <iostream>
 8 #include <exception>
 9
10 #include "Lista.hpp"
11 #include "Movie.hpp"
12 #include "FamilyMovie.hpp"
13 #include "Documentary.hpp"
14 #include "test.hpp"
15 #include "gtest_lite.h"
16
17 using std::cout;
18 using std::cin;
19 using std::cerr;
20 using std::endl;
21 using std::bad_alloc;
22
23 #define TESZT
24
28 void options();
29
35 unsigned int choice(Lista l);
36
40 int main(){
41
42     try{
43         Lista list;
44
45         list.read("film_database.txt");
46
47         int valasztas;
48         string valasz;
49
50         string mufaj, leiras = "", cim;
51         unsigned int hossz, kiadas, korhatar = 0;
52
53         do
54         {
55             options();
56             valasztas = choice(list);
57
58         switch(valasztas){
59
```

```
60              case 1:
61
62                  list.print();
63
64                  break;
65
66              case 2:
67
68                  do{
69
70                      cin.ignore();
71                      cout << "Mufaj: ";
72                      getline(cin, mufaj);
73                      cout << endl;
74                      cout << "Cim: ";
75                      getline(cin, cim);
76                      cout << endl;
77                      cout << "Hossz (perc): ";
78                      cin >> hossz;
79
80                      if(cin.fail()){
81
82                          list.save("film_database.txt");
83                          list.felszab();
84                          throw "\n\nHibas bemenet. A program leall.\n";
85                      }
86
87                      cout << endl;
88                      cout << "Kiadas: ";
89                      cin >> kiadas;
90
91                      if(cin.fail()){
92
93                          list.save("film_database.txt");
94                          list.felszab();
95                          throw "\n\nHibas bemenet. A program leall.\n";
96                      }
97
98                      if(mufaj == "csaladi"){
99                          cout << endl;
100                         cout << "Korhatar: ";
101                         cin >> korhatar;
102
103                         if(cin.fail()){
104
105                             list.save("film_database.txt");
106                             list.felszab();
107                             throw "\n\nHibas bemenet. A program leall.\n";
108                         }
109
110                         cout << endl;
111
112                         list.add(mufaj, cim, hossz, kiadas, korhatar, leiras);
113
114                     }
115                     else{
116                         if(mufaj == "dokumentum"){
117                             cout << endl;
118                             cin.ignore();
119                             cout << "Leiras: ";
120                             getline(cin, leiras);
121                             cout << endl;
122
123                             list.add(mufaj, cim, hossz, kiadas, korhatar, leiras);
124                         }
125                         else{
126                             cout << endl;
127                             list.add(mufaj, cim, hossz, kiadas, korhatar, leiras);
128                         }
129                     }
130
131                 cout << "Szeretne tovabbi filmeket hozzaadni? ";
132                 cin >> valasz;
133                 cout << endl;
134
135                 }while(valasz == "igen");
136
137                 break;
138
139             case 3:
140
```

```
141                 do{
142                     cin.ignore();
143 
144                     cout << "A torlendo film cime: ";
145                     getline(cin, cim);
146 
147                     cout << endl;
148 
149                     list.remove(cim);
150 
151                     cout << "\nSzeretne tovabbi filmeket torolni? ";
152                     cin >> valasz;
153                     cout << endl;
154 
155                 }while(valasz == "igen");;
156 
157                 break;
158 
159             case 4:
160 
161                 cin.ignore();
162 
163                 cout << "Mely mufaj szerint tortenjen a szures? ";
164                 getline(cin, mufaj);
165                 cout << endl;
166 
167                 list.filter(mufaj);
168 
169                 break;
170 
171             case 5:
172 
173                 cin.ignore();
174 
175                 cout << "A keresett film cime: ";
176                 getline(cin, cim);
177                 cout << endl;
178 
179                 list.search(cim);
180 
181                 break;
182 
183             default:
184                 break;
185             }
186 
187         }while(valasztas != 6);
188 
189         if(list.get_eleje() != nullptr){
190             list.save("film_database.txt");
191             list.felszab();
192         }
193 
194         cout << "Sikeres kilepes, viszontlatasra!" << endl;
195 
196     }catch(bad_alloc& ba){
197 
198         cerr << "\nHiba tortent memoriafoglalas kozben. A program leall.\n" << endl;
199         cerr << ba.what();
200         exit(1);
201     }
202     catch(const char* p){
203         cerr << p;
204         exit(1);
205     }
206     catch(...){
207         cerr << "\nVaratlan kivetel keletkezett. A program leall.\n" << endl;
208         exit(1);
209     }
210 
211     #ifdef TESZT
212         try{
213             test_0();
214             test_1();
215             test_2();
216             test_3();
217             test_4();
218             test_5();
219             test_6();
220             test_7();
221         }
```

```
222         catch(...){
223             cerr << "\nVaratlan kivetel keletkezett teszteles soran. A program leall.\n" << endl;
224             exit(1);
225         }
226     #endif
227 
228     return 0;
229 }
230 
231 void options(){
232 
233     cout << "FILMTAR\n" << endl;
234     cout << "1. Teljes adatbazis megjelenitese" << endl;
235     cout << "2. Uj film(ek) felvetele" << endl;
236     cout << "3. Meglevo film(ek) torlese" << endl;
237     cout << "4. Mufaj szerinti szures" << endl;
238     cout << "5. Kereses" << endl;
239     cout << "6. Kilepes\n" << endl;
240 }
241 
247 unsigned int choice(Lista l){
248 
249     unsigned int valasztas;
250 
251     cout << "Kerem valasszon a menubol (1-6): ";
252     cin >> valasztas;
253 
254     if(cin.fail()){
255         l.felszab();
256         throw "\n\nHibas bemenet. A program leall.\n";
257     }
258 
259     cout << endl;
260 
261     while(valasztas == 0 || valasztas > 6)
262     {
263         cout << "A kivalasztott menupont nem letezik!\n\n";
264 
265         cout << "Kerem valasszon ujra a menubol (1-6): ";
266         cin >> valasztas;
267 
268         if(cin.fail()){
269             l.felszab();
270             throw "\n\nHibas bemenet. A program leall.\n";
271         }
272 
273         cout << endl;
274     }
275 
276     return valasztas;
277 }
```

---

files/Movie.hpp fájlreferencia

Osztályok

- class Movie - *Movie alaposztály.*

Részletes leírás

Ez a fájl tartalmazza a Movie ősosztály deklarációját, illetve az osztály tagfüggvényeinek definícióját.

(Definíció a(z) Movie.hpp fájlban.)

Movie.hpp

```
  1 #ifndef MOVIE_HPP
  2 #define MOVIE_HPP
  3 
 10 #include <iostream>
 11 #include <fstream>
 12 
 13 using std::string;
 14 using std::ofstream;
 15 
```

```
   17 class Movie{
   18
   19     string mufaj, cim;
   20     unsigned int hossz, kiadas;
   21
   22     public:
   23
   29         explicit Movie(string m = "", string c = "", unsigned int h = 0, unsigned int k = 0) :
mufaj(m), cim(c), hossz(h), kiadas(k) {}
   30
   32         virtual ~Movie() {}
   33
   36         string get_cim()const{return cim;}
   37
   40         string get_mufaj()const{return mufaj;}
   41
   44         unsigned int get_hossz()const{return hossz;}
   45
   48         unsigned int get_kiadas()const{return kiadas;}
   49
   51         virtual void show(){
   52             std::cout << mufaj  << " " << cim << " " << hossz << " " << kiadas;
   53         }
   54
   57         virtual void print_in_file(ofstream& f){
   58             f << mufaj << "\t" << cim << "\t" << hossz << "\t" << kiadas << "\n";
   59         }
   60 };
   61
   62 #endif // MOVIE_HPP
```

---

files/test.cpp fájlreferencia

Részletes leírás

Ez a fájl tartalmazza a teljes adatbázis teszteléséhez szükséges függvények definícióját. Fontos, hogy a tesztelő függvények a megadott és feltöltött teszt adatbázissal (test_database.txt) operálnak. Helyes működésük csak ennek használatával garantált. A tesztesetek csak abban az estben hajtódnak végre, ha a main.cpp fájlban a TESZT makró definiálva van. A listából való törlés és a felszabadító függvény esetén a dinamikus memória felszabadításának sikerességét a Memtrace nevű, inkludált eszköz végzi, és a program végrehajtása után hibaüzenetet ad, amennyiben nem megfelelő működést tapasztalt.

(Definíció a(z) test.cpp fájlban.)

Függvények dokumentációja

int count (ListaElem * *eleje*)

A láncolt lista elemeinek megszámlálására szolgáló függvény.

A függvény végigiterál a láncolt listán, és minden listaelemnél egy alapértelmezetten lenullázott számlálóhoz hozzáad egyet.

(Definíció a(z) test.cpp fájlban.)

int filter_count (ListaElem * *eleje*, string *h*)

Megszámlálja a listában található, paraméterként megadott műfajjal megegyező műfajú filmeket.

A függvény megegyezik a Lista.cpp fájlban definiálttal. Az egyetlen különbség, hogy jelen esetben a függvény megszámlálja a megadott műfajjal megegyező műfajú filmeket, ahelyett, hogy kiírná azok adatait.

(Definíció a(z) test.cpp fájlban.)

int search_count (ListaElem * *eleje*, string *o*)

Megszámlálja a listában található, paraméterként megadott címmel megegyező című filmeket.

A függvény megegyezik a Lista.cpp fájlban definiálttal. Az egyetlen különbség, hogy jelen esetben a függvény megszámlálja a megadott címmel megegyező című filmeket, ahelyett, hogy kiírná azok adatait.

(Definíció a(z) test.cpp fájlban.)

void test_0 ()

A Lista osztály alapértelmezett konstruktorának tesztelése.

Ellenőrizzük, hogy ősosztály, illetve a két származtatott osztály konstruktora helyesen működik-e.

(Definíció a(z) test.cpp fájlban.)

void test_1 ()

A fájlból való beolvasás sikerességének tesztelése.

Beolvasás után megszámláljuk, hogy mennyi elem található a listában. Ha az megegyezik a fájlban található filmek számával, akkor a tesztet sikeresnek tekintjük.

Definíció a(z) test.cpp fájlban.)

void test_2 ()

A listához való hozzáadás sikerességének tesztelése.

Létrehozunk egy ideiglenes listát, amihez hozzáadunk 3 filmet, majd megszámláljuk a létrejött láncolt lista elemeinek számát. Ezenkívül leellenőrizzük, hogy valóban azok az értékek kerültek-e be a listába, amiket szerettünk volna. Ha mindkét ellenőrzés sikeres, akkor a tesztet sikeresnek tekintjük.

(Definíció a(z) test.cpp fájlban.)

void test_3 ()

A listából való törlés sikerességének tesztelése.

Létrehozunk egy ideiglenes listát, amihez hozzáadunk 3 filmet, majd leellenőrizzük a 3 lehetséges törlési lehetőséget a megmaradt elemek értékeinek ellenőrzésével.

(Definíció a(z) test.cpp fájlban.)

void test_4 ()

A megadott műfajú filmek kiszűrésének tesztelése.

A módosított visszatérési értékű szűrő függvény segítségével megszámláljuk, hogy hány, a megadott műfajjal megegyező műfajú film szerepel a listában. Ha a kapott érték megegyezik az elvárttal, akkor a tesztet sikeresnek tekintjük.

(Definíció a(z) test.cpp fájlban.)

void test_5 ()

Megadott című filmek keresésének tesztelése.

A módosított visszatérési értékű kereső függvény segítségével megszámláljuk, hogy hány, a megadott címmel megegyező című film szerepel a listában. Ha a kapott érték megegyezik az elvárttal, akkor a tesztet sikeresnek tekintjük.

(Definíció a(z) test.cpp fájlban.)

void test_6 ()

A fájlba való kiíratás sikerességének tesztelése.

Egy ideiglenes listába beolvassuk a test_database.txt fájl tartalmát, majd hozzáadunk plusz egy elemet. Ezután a listát visszamentjük a test_database.txt fájlba. Ha a következő beolvasásnál egyel több eleme lesz a listának, mint korábban, akkor a tesztet sikeresnek tekintjük.

(Definíció a(z) test.cpp fájlban.)

void test_7 ()

A read() függvény kivétel dobásának ellenőrzése.

Létrehozunk egy ideiglenes üres listát, majd megpróbáljuk ebbe a listába egy nem létező fájl tartalmát beolvasni. Ha a read() függvény kivételt dob, akkor a tesztet sikeresnek tekintjük.

(Definíció a(z) test.cpp fájlban.)

test.cpp

```
 1
10 #include "gtest_lite.h"
11 #include "test.hpp"
12
13 using std::cout;
14 using std::endl;
15
19 int count(ListaElem *eleje){
20
21     int db = 0;
22     ListaElem *aktualis = eleje;
23
24     while(aktualis != nullptr)
25     {
26         db += 1;
27         aktualis = aktualis ->kov;
28     }
29
30     return db;
31 }
32
37 int filter_count(ListaElem* eleje, string h){
38
39     int db = 0;
40
41     ListaElem* temp;
42     for(temp = eleje; temp; temp = temp->kov){
43         if(temp->fp->get_mufaj() == h){
44            ++db;
45         }
46     }
47
48     return db;
49 }
50
55 int search_count(ListaElem* eleje, string o){
56
57     int db = 0;
58
59     ListaElem* temp;
60     for(temp = eleje; temp; temp = temp->kov){
61         if(temp->fp->get_cim() == o){
62             ++db;
63         }
64     }
65
```

```
   66     return db;
   67 }
   68 
   72 void test_0(){
   73 
   74     TEST(Test_0, konstruktor){
   75         Lista l2;
   76         EXPECT_EQ(l2.get_eleje(), nullptr) << "Nem jol mukodik a default konstruktor." << endl;
   77 
   78         Movie m;
   79         EXPECT_EQ("", m.get_mufaj()) << "Nem jol mukodik a default konstruktor (Movie)." << endl;
   80         EXPECT_EQ("", m.get_cim()) << "Nem jol mukodik a default konstruktor (Movie)." << endl;
   81         EXPECT_EQ(0, static_cast<int>(m.get_hossz())) << "Nem jol mukodik a default konstruktor
(Movie)." << endl;
   82         EXPECT_EQ(0, static_cast<int>(m.get_kiadas())) << "Nem jol mukodik a default konstruktor
(Movie)." << endl;
   83 
   84         FamilyMovie fm;
   85         EXPECT_EQ("", fm.get_mufaj()) << "Nem jol mukodik a default konstruktor (FamilyMovie)." <<
endl;
   86         EXPECT_EQ("", fm.get_cim()) << "Nem jol mukodik a default konstruktor (FamilyMovie)." << endl;
   87         EXPECT_EQ(0, static_cast<int>(fm.get_hossz())) << "Nem jol mukodik a default konstruktor
(FamilyMovie)." << endl;
   88         EXPECT_EQ(0, static_cast<int>(fm.get_kiadas())) << "Nem jol mukodik a default konstruktor
(FamilyMovie)." << endl;
   89         EXPECT_EQ(0, static_cast<int>(fm.get_korhatar())) << "Nem jol mukodik a default konstruktor
(FamilyMovie)." << endl;
   90 
   91         Documentary d;
   92         EXPECT_EQ("", d.get_mufaj()) << "Nem jol mukodik a default konstruktor (Documentary)." << endl;
   93         EXPECT_EQ("", d.get_cim()) << "Nem jol mukodik a default konstruktor (Documentary)." << endl;
   94         EXPECT_EQ(0, static_cast<int>(d.get_hossz())) << "Nem jol mukodik a default konstruktor
(Documentary)." << endl;
   95         EXPECT_EQ(0, static_cast<int>(d.get_kiadas())) << "Nem jol mukodik a default konstruktor
(Documentary)." << endl;
   96         EXPECT_EQ("", d.get_leiras()) << "Nem jol mukodik a default konstruktor (Documentary)." <<
endl;
   97 
   98     }END
   99     cout << endl;
  100 }
  101 
  105 void test_1(){
  106 
  107     TEST(Test_1, beolvasas){
  108         Lista l2;
  109 
  110         EXPECT_NO_THROW(l2.read("test_database.txt")) << "A read() fuggveny kivetelt dobott." << endl;
  111 
  112         int elvart = 25;
  113         int kapott = count(l2.get_eleje());
  114 
  115         EXPECT_EQ(elvart, kapott) << "Nem sikerult a fajlbol valo beolvasas." << endl;
  116 
  117         l2.felszab();
  118     }END
  119     cout << endl;
  120 }
  121 
  126 void test_2(){
  127 
  128     TEST(Test_2, hozzaadas){
  129 
  130         Lista l2;
  131         l2.add("csaladi", "Fogoly", 55, 1999, 12, "");
  132         l2.add("dokumentum", "Gogol", 85, 2000, 0, "Gogol elete");
  133         l2.add("vigjatek", "Lobonc", 105, 2001, 0, "");
  134 
  135         int elvart = 3;
  136         int kapott = count(l2.get_eleje());
  137         EXPECT_EQ(elvart, kapott) << "Nem sikerult a listahoz valo hozzaadas. 0" << endl;
  138 
  139         EXPECT_EQ("Fogoly", l2.get_eleje()->fp->get_cim()) << "Nem sikerul a listahoz valo hozzaadas.
1" << endl;
  140         EXPECT_EQ("dokumentum", l2.get_eleje()->kov->fp->get_mufaj()) << "Nem sikerul a listahoz valo
hozzaadas. 2" << endl;
  141         EXPECT_EQ(105, static_cast<int>(l2.get_eleje()->kov->kov->fp->get_hossz())) << "Nem sikerült a
listahoz valo hozzaadas. 3"; // Castolni kell, különben unsigned <--> int.
  142 
  143 
  144         l2.felszab();
```

```
145     }END
146     cout << endl;
147 }
148
152 void test_3(){
153
154     TEST(Test_3, torles){
155
156         Lista l2;
157         l2.add("csaladi", "Fogoly", 55, 1999, 12, "");
158         l2.add("dokumentum", "Gogol", 85, 2000, 0, "Gogol elete");
159         l2.add("vigjatek", "Lobonc", 105, 2001, 0, "");
160
161         l2.remove("Fogoly");    // Törlés az elejérõl.
162         EXPECT_EQ("dokumentum", l2.get_eleje()->fp->get_mufaj()) << "Hibas torles (eleje)." << endl;
163         EXPECT_EQ(2, count(l2.get_eleje())) << "A torles nem sikerult (eleje)." << endl;
164
165         l2.remove("Lobonc");    // Törlés a végérõl.
166         EXPECT_EQ(l2.get_eleje()->kov, nullptr) << "Hibas torles (vege)." << endl;
167         EXPECT_EQ(1, count(l2.get_eleje())) << "A torles nem sikerult (vege)." << endl;
168
169         l2.add("csaladi", "Fogoly", 55, 1999, 12, "");
170         l2.add("vigjatek", "Lobonc", 105, 2001, 0, "");
171
172         l2.remove("Fogoly");    // Törlés a közepérõl.
173         EXPECT_EQ(105, static_cast<int>(l2.get_eleje()->kov->fp->get_hossz())) << "Hibas torles
(kozepe)." << endl; // Castolni kell, különben unsigned <--> int.
174         EXPECT_EQ(2, count(l2.get_eleje())) << "A torles nem sikerult (kozepe)." << endl;
175
176         l2.felszab();
177     }END
178     cout << endl;
179 }
180
185 void test_4(){
186
187     TEST(Test_4, szures){
188
189         Lista l2;
190
191         EXPECT_NO_THROW(l2.read("test_database.txt")) << "A read() fuggveny kivetelt dobott" << endl;
192
193         int elvart = 7; // Akcio filmek száma.
194         int kapott = filter_count(l2.get_eleje(), "akcio");
195         EXPECT_EQ(elvart, kapott) << "Nem sikerult megtalalni az osszes filmet." << endl;
196
197         l2.felszab();
198     }END
199     cout << endl;
200 }
201
206 void test_5(){
207
208     TEST(Test_5, kereses){
209
210         Lista l2;
211
212         EXPECT_NO_THROW(l2.read("test_database.txt")) << "A read() fuggveny kivetelt dobott." << endl;
213
214         int elvart = 2; // "A vonat" címû filmek száma.
215         int kapott = search_count(l2.get_eleje(), "A vonat");
216         EXPECT_EQ(elvart, kapott) << "Nem sikerult az osszes megadott cimu filmet megtalalni." << endl;
217
218         l2.felszab();
219     }END
220     cout << endl;
221 }
222
227 void test_6(){
228
229     TEST(Test_6, kiiratas){
230
231         Lista l2;
232
233         EXPECT_NO_THROW(l2.read("test_database.txt")) << "A read() fuggveny kivetelt dobott." << endl;
234
235         l2.add("alma", "korte", 120, 2006, 0, "");
236
237         l2.save("test_database.txt");
238
239         l2.felszab();
```

```
240
241         EXPECT_NO_THROW(l2.read("test_database.txt")) << "A read() fuggveny kivetelt dobott." << endl;
242
243         int db = count(l2.get_eleje());
244         EXPECT_EQ(26, db) << "Hibas kiiratas fajlba." << endl;
245
246         l2.remove("korte");
247
248         l2.save("test_database.txt");
249
250         l2.felszab();
251
252     }END
253
254     cout << endl;
255 }
256
260 void test_7(){
261
262     TEST(Test_7, read_throw){
263
264         Lista l2;
265         try{
266             l2.read("nonexistent.txt");
267
268             EXPECT_THROW(cout << "Hiba tortent az adatbazis beolvasasa soran. A program kilep.", const
char*) << "A read() fuggveny nem dobott kivetelt." << endl;
269         }
270         catch(const char*){}
271
272     }END
273     cout << endl;
274 }
```

---

files/test.hpp fájlreferencia

Függvények

- int count (ListaElem *eleje)
  A láncolt lista elemeinek megszámlálására szolgáló függvény.
- int filter_count (ListaElem *eleje, string h)
  Megszámlálja a listában található, paraméterként megadott műfajjal megegyező műfajú filmeket.
- int search_count (ListaElem *eleje, string o)
  Megszámlálja a listában található, paraméterként megadott címmel megegyező című filmeket.
- void test_0 ()
  A Lista osztály alapértelmezett konstruktorának tesztelése.
- void test_1 ()
  A fájlból való beolvasás sikerességének tesztelése.
- void test_2 ()
  A listához való hozzáadás sikerességének tesztelése.
- void test_3 ()
  A listából való törlés sikerességének tesztelése.
- void test_4 ()
  A megadott műfajú filmek kiszűrésének tesztelése.
- void test_5 ()
  Megadott című filmek keresésének tesztelése.
- void test_6 ()
  A fájlba való kiíratás sikerességének tesztelése.
- void test_7 ()
  A read() függvény kivétel dobásának ellenőrzése.

Részletes leírás

Ez a fájl tartalmazza a teszteléshez szükséges függvények deklarációit.

(Definíció a(z) test.hpp fájlban.)