Programozói dokumentáció

Könyvtár

Fordítási környezet:

1. A konyvtar.exe program futtatásához szükséges a MinGW GCC compiler
2. Szabványos függvénykönyvtárok

Fordítás folyamata:

1. Létre kell hozni a "filementes.o", " fileolvasas.o", "kezelofelulet.o", "kolcsonzesek_felhasznaloknallevo.o", " konyv_kezeles.o", " Lista_modositas.o", " stringkezeles.o", "main.o" nevű fájlokat a parancssorban:
   gcc -c file_nev.c -o file_nev.o
2. Létre kell hozni a konyvtar.exe fájlt:
   gcc f ilementes.o fileolvasas.o … main.o -o konyvtar.exe

Könyvtár:

Ez a program egy könyvtári adatbázis kezelésére lett létrehozva. Az adatok átlátható kezelése, és az azokkal való könnyű dolgozás érdekében a program menüpontokra lett bontva, van amelyik a meglévő adatbázis átalakítására lett létrehozva, és van olyan amelyik a meglévő adatok közötti keresésre lett létrehozva.

A meglévő adatok átalakítására 3 alcsoportra lettek bontva, új adatok hozzáadására, meglévő adatok módosítására, és meglévő adatok törlésére.

A keresési opcióknál létre lett hozva egy univerzális keresési funkció, amelyben az adatstruktúra bármely paraméterére rá lehet keresni. Ezen kívül van egy keresési fül, ami direkt módon megadja, hogy az adott könyv ki által van kikölcsönözve (könyv címe alapján való keresés). A másik keresési opció pedig ennek az ellenkezője: megadja, hogy az adott kölcsönzőnél éppen melyik könyvek találhatók.

Adatszerkezet választása:

A program fő adatszerkezetét az "Adatok" láncolt lista alkotja, ez tárolja el az adatbázisból érkező összes adatot. A lista nem tartalmaz strázsát, ez egy egyszerű, egyszeresen láncolt lista, amiben az elemek a beolvasás sorrendjében vannak eltárolva.

A lista tartalmazza a könyvek címét, szerzőjét, témáját, kiadási évét (pozitív egész szám), és hogy kinél van az adott könyv. A szövegrészeket tartalmazó elemek hosszúságát előre nem lehet eldönteni, ezért azokat (cím, szerző, téma, kölcsönző) nem fix elemszámú karaktertömbökként vannak definiálva. Ezeket az adatokat majd a beolvasásnál dinamikusan fogjuk lefoglalni.

```
typedef struct Adatok
{
    char *book_title;
    char *author;
    char *theme;
    int  release_year;
    char *borrower;
    struct Adatok *next;    /// Lancolt lista letrehozasa
}Adatok;
```

A programhoz tartozó adatbázisát a "konyvtar.txt" alkotja. Itt a könyvek öt adata ";"-vel van elválasztva és egy sor egy könyv adatait tárólja. Beolvasáskór és mentéskor is ezt a fájlt használja a program.

```
Vuk;Istvan Fekete;Adventure;1965;Ved Elek
```

A program működését vezérlő modulok:

1. **Kezelőfelület:** (kezelofelulet.c)
   A kezelőfelület alkotja a program főmenüjét, innen lehet a programot vezérelni.
2. **Fájl beolvasása:** (fileolvasas.c)
   Itt olvassa be a program a "konyvtar.txt"-ből az adatok a listába.
3. **Fájl mentése:** (filementes.c)
   A program itt menti ki a módosított listát a "konyvtar.txt"-ba.
4. **Kölcsönzések:** (kolcsonzesek_felhasznaloknallevo.c)
   Itt találhatók a kölcsönzések és felhasználók alapján való kereső függvények
5. **Könyvek kezelése:** (konyv_kezeles.c)
   Itt találhatók az új adatelemek létrehozásáért, a meglévők módosításáért és a törlésért felelős függvények.
6. **Láncolt lista kezelése:** (Lista_modositas.c)
   Itt találhatók a láncolt lista kezelésére szolgáló függvények.
7. **Szöveg beolvasása:** (stringkezeles.c)
   Itt található a szöveg beolvasásáért felelős függvény.

A program működését vezérlő függvények:

<u>Beolvasásért felelős függvények:</u>

```
Adatok *konyvtari_adatok()
```

A függvény létrehoz egy lista elemet, ami kezdetben NULL, majd megnyitja az adatokat tartalmazó "konyvtar.txt", ha a fájl létezik, akkor továbbadja a soronként beolvasó függvénynek, ha végzett a teljes beolvasással, akkor a "konyvari_adatok" függvény visszaad egy láncolt listát. Ha a fájl nem létezik, akkor egy üres listát és egy hibakódot ad vissza a függvény.

```
Adatok *file_beolvasas(FILE *fp)
```

A függvény paraméterként megkapja a megnyitott fájlt, majd azt sorokra bontja, amíg a sor értéke nem NULL. Ha ez sikeresen megtörtént, akkor visszaad egy listát a "konyvari_adatok"-nak. A sorok hossza legfeljebb 256 karakter lehet, de ezen belül az egyes adatok hossza eltérő lehet. Ha a sor nem NULL, akkor a függvény létrehoz egy listaelemet, amibe a "valogato" függvény segítségével feltölti azt, majd a "lista_hozzafuz" segítségével hozzáadja azt a "betolto" listához, ami a függvény visszatérési értéke lesz.

```
Adatok *valogato (char *sor)
```

A függvény paraméterként kap egy egész sort a "file_beolvasas" sorokra bontó ciklusától, amit a függvény tovább bontja az adatstruktúra elemeire, úgy hogy megkeresi a szövegben lévő ";"-et és azok szerint résszövegekre bontja, amit egy 5 elemű karaktertömbben tárol el ("char *stringek[5]"), mivel az 5. elemnek a végén a

sor beolvasásakor még megmaradt a "\n" karakter, ezért azt a "kolcsonzo" függvény segítségével leválasztjuk. Miután az elemek szét lettek válogatva, azokat betöltjük egy listaelembe (az egyes szövegrészeket a listában dinamikusan foglaljuk le, mivel előre nem tudjuk azok hosszúságát, ezeket majd a lista felszabadításkor is külön fel kell szabadítani), ami a függvény visszatérési értéke lesz.

```
char *kolcsonzo (char *borrower)
```

A függvény paraméterként megkapja a "valogato"-tól a sor utolsó elemét, majd arról leválasztja annak utolsó karakterét ("\n"), és visszatér ezzel az újonnan létrehozott "stringgel".

Mentésért felelős függvények:

```
void fileba_mentes (Adatok *lista)
```

A függvény paraméterként kap egy listát, majd azok elemeit elmenti a "konyvtar.txt" fájlba az alábbi formátumban: "cím;szerzo;tema;kiadasi_ev;kolcsonzo\n".

Kezelőfelület:

```
void kezelofelulet ()
```

A függvény a főmenü konzolban való megjelenítéséért felelős, létrehoz egy egész szám változót, amivel a menüpontok között lehet választani, ha a beolvasott fájl nem létezik vagy üres, akkor jelzi a felhasználónak, hogy hozzon létre legalább egy elemet amivel a program tud dolgozni.

```
void konyvek ()
```

Ebbe a függvénybe lép át a főmenü, ha a felhasználó a könyvek kezelése menüpontot választja. Itt lehet kiválasztani az elemet hozzáadása, módosítása, vagy törlése menüpontokat. Felépítése hasonló a "kezelofelulet"-hez.

```
void kereso ()
```

Ebbe a függvénybe lép át a főmenü, ha a felhasználó a keresés menüpontot választja. Itt lehet kiválasztani azt, hogy melyik adat alapján szeretnénk keresni. Felépítése hasonló a "kezelofelulet"-hez.

Kölcsönzések és felhasználónál lévő könyvek:

```
void kolcsonzesek ()
```

Ebbe a függvénybe lép át a főmenü, ha a felhasználó a "Kolcsonzesek" menüpontot választja. A függvény bekéri a könyv címét a felhasználótól ("string_in" függvénnyel), majd összeveti azt a listában található címekkel ("kikolcsonzott_konyvek" függvény), ha van egyezés, akkor kiírja, hogy ki kölcsönözte ki az adott könyvet, ha nincs egyezés, akkor hibaüzenetet ad.

```
void felhasznaloi ()
```

Ebbe a függvénybe lép át a főmenü, ha a felhasználó a "Felhasznalonal levo konyvek" menüpontot választja. A függvény bekéri a kölcsönző nevét a felhasználótól ("string_in" függvénnyel), majd összeveti azt a listában található címekkel

("felhasznalonal_levo_konyvek" függvény), ha van egyezés, akkor kiírja a felhasználónál lévő könyvek összes adatát, ha nincs egyezés, akkor hibaüzenetet ad.

Mindkét függvényben először a program létrehoz egy listát, amibe beletölti a beolvasott adatokat, és utolsó lépésként felszabadítja azt, és a felhasználó által megadott adatokat.

Könyvek kezelése:

```
void letrehoz ()
```

Ebbe a függvénybe lép át a "konyvek" függvény, ha a felhasználó a "Uj letrehozasa" menüpontot választja. A függvény bekéri a könyv összes adatát a felhasználótól ("string_in" függvénnyel), majd hozzáfűzi azt a listához a "lista_hozzafuz" függvény segítségével, és elmenti azt.

```
void torles ()
```

Ebbe a függvénybe lép át a "konyvek" függvény, ha a felhasználó a "Meglevo torlese" menüpontot választja. A függvény bekéri a könyv összes adatát a felhasználótól ("string_in" függvénnyel), majd meghatározza azok indexét a listában egy függvény segítségével, ha a kapott érték -1, akkor az azt jelenti, hogy nincs a listában, emellett megnézi azt is, hogy az input "*" jel-e, ha igen, akkor a program úgy értelmezi, hogy ott bármely listaelem állhat, de az összes adat helyén nem állhat "*". Ha a program talál egyezést, akkor egy elem törlő függvény segítségével törli azt a listából, ha nincs egyezés, akkor hibaüzenetet ad.

```
void modosit ()
```

Ebbe a függvénybe lép át a "konyvek" függvény, ha a felhasználó a "Meglevo modositasa" menüpontot választja. A függvény bekéri a könyv összes adatát a felhasználótól ("string_in" függvénnyel), majd meghatározza azok indexét a listában egy függvény segítségével, ha a kapott érték -1, akkor az azt jelenti, hogy nincs a listában, emellett megnézi azt is, hogy az input "*" jel-e, ha igen, akkor a program úgy értelmezi, hogy ott bármely listaelem állhat, de az összes adat helyén nem állhat "*". Ha a program talál egyezést, akkor új adatokat kér be a felhasználótól, amire kicseréli a régi adatokat, ha nincs egyezés, akkor hibaüzenetet ad.

```
int ev (char *rls_yr)
```

Mind a módosítás, mind a törlés függvényekben szerepel a könyv kiadási éve, azonban ezeket is karakterként olvassa be a függvény, mivel itt is szerepelhet "*", ezért a függvény paraméterként átvesz egy stringet. Annak érdekében, hogy ne legyen típus eltérési hiba ezt át kell alakítani egész számmá, így egy könyvtári függvény segítségével megnézi a program, hogy az input "*"-e, ha igen, akkor -1-et ad vissza, ha nem akkor átalakítja azt egész számmá és visszatér azzal.

A "torles", "modosit", "letrehoz" függvényekben először a program létrehoz egy listát, amibe beletölti a beolvasott adatokat, és utolsó lépésként felszabadítja azt, és a felhasználó által megadott adatokat.

Láncolt lista kezelése:

```
void felszabadit (Adatok *lista)
```

A függvény felszabadítja a paraméterként kapott lista elemeit. Mivel az "Adatok" struktúra elemei között vannak dinamikusan foglalt elemek, ezért azokat a függvényen belül külön fel kell szabadítani (pl.: free(lista->author)) azokat, majd a legvégén magát a listaelemet kell felszabadítani (pl.: free(lista)). A felszabadító ciklust addig léptetjük, amíg az adott lista elem nem NULL.

```
Adatok *lista_hozzafuz (Adatok *lista, Adatok *hozzafuzendo)
```

A függvény paraméterként kap egy listát, és egy lista elemet, amit hozzáfűz a lista végéhez.

```
void lista_kiir (Adatok *lista)
```

A függvény kiirja a paraméterként kapott lista elemeit.

```
int lista_elem_keres_szoveg (Adatok *lista, char *szoveg)
```

Megadja a keresett szöveg indexét a listában, ha a szöveg nem található benne, akkor -1-et ad vissza

```
int lista_elem_keres_ev (Adatok *lista, int szam)
```

Megadja a keresett évszám indexét a listában, ha az nincs a listában, akkor -1-et ad vissza.

```
Adatok *elem_torlese (Adatok *lista, int hanyadik)
```

Paraméterként átvesz egy listát, és egy egész számot, ami a lista egy adott elemének indexét jelenti, majd törli azt, és visszatér a hátramaradó listával.

```
void felhasznaloknal_levo_konyvek (Adatok *lista, char *nev)
```

A függvény összeveti a kapott nevet a listában található kölcsönzök nevével, ha van egyezés, akkor kiírja a könyv adatait.

```
void kikolcsonzott_konyvek (Adatok *lista, char *cim)
```

A függvény összeveti a kapott könyv címeket a listában található könyvek címével, ha van egyezés, akkor kiírja a kölcsönző nevét.

```
Adatok *elem_letrehoz ()
```

Létrehoz a függvényen belül egy listaelemet, majd visszatér vele.

```
Adatok *lista_atlancol (Adatok *lista, int hanyadik)
```

A függvény az adott indexű helyre befűz egy új elemet, amit a függvényen bellül kér be a felhasználótól, és az ott található régit felszabadítja.

```
void lista_konnyv_kereso ()
```

A függvény bekér egy szöveget a felhasználótól, majd megnézi, hogy az megtalálható-e a listában, ha igen akkor kiírja az egyező könyv adatait, ha nem hibaüzenetet ad.

```
void lista_konnyv_kereso_ev ()
```

A függvény bekér a felhasználótól egy évszámot, majd megnézi, hogy az megtalálható-e a listában, ha igen akkor kiírja az egyező könyv adatait, ha nem hibaüzenetet ad.

Stringkezelés:

```
char string_in ()
```

A függvény karaktereket olvas be "\n"-ig majd visszaad egy string-et.