Programozói dokumentáció

1. A feladat

A program egy iskolai könyvtár-rendszer feladatainak ellátását kell elvégeznie. A feladatok a következők:

- Könyvek és diákok adatainak tárolása. A program ezt két különböző (Book_Records.txt, Student_Records.txt) fájlban tárolja.
- Új könyvek felvétele, könyvek törlése, kilistázása, könyvek keresése, kiadása egy diáknak.
- Diákok adatainak felvétele, változásra hajlamos adatainak (telefonszám) módosítása és diákoknál lévő könyvek kilistázása.

A program különböző modulokra van bontva, melyek adott feladatcsoportok ellátására képesek.

2. Adatok feldolgozása

Mivel két különböző felépítésű fájlban tárolódnak az adatok, ezért ezek feldolgozását egy-egy láncolt lista segíti. Ezek segítségével könnyen végezhető a keresés, elem törlése, új elem hozzáfűzése és a fájlból való beolvasás. Ezen szempontok miatt esett erre a választás.

A könyvek és diákok feldolgozásához struktúrák vannak létrehozva, melyek egyben a rendezetlen láncolt lista elemei is

- A bookdata_t struktúra elemei:
  - int id (->egyedi azonosító (szám típus))
  - char *title (->könyv címe (karaktertömb típus))
  - char *autname (->könyv írója (karaktertömb típus))
  - int year (->kiadásának éve (szám típus))
  - char *category (->kategóriája (karaktertömb típus))
  - int where (->helyzete: ha a könyvtárban van akkor 0, ha diáknál akkor a diák egyedi azonosítója (szám típus))
- A bookdata_t struktúra elemei:
  - int id (->egyedi azonosító (szám típus))
  - char *name (->diák neve (karaktertömb típus))
  - char *phonenum (->diák telefonszáma (karaktertömb típus), így mindkét telefonszám formátum elfogadott)

3. A *.txt fájlok

Az adatok pontosvesszővel vannak elválasztva a fájlokban, sorok végén viszont csak ENTER van

- Book_Records.txt
  - A könyvek adatainak tárolására szolgál
  - Adatok:
    - könyv egyedi azonosítója; könyv címe; könyv írója; kiadási év; kategória; könyv jelen helyzete (ha 0, akkor a könyvtárban van, egyébként a diák azonosítója akinél van)
    - 27;The Art of War;Sun Tzu;2020;History;0
    - 3;Ansible for DevOps;Jeff Geerling;2008;Computers and Programming;2
  - Felhasználó a programon keresztül tud új adatsort írni, kitörölni egy adatsort és módosítani egy adatsor adatain

- Student_Records.txt
    - Diákok adatainak tárolására szolgál
    - Adatok:
        - diák egyedi azonosítója; diák neve; diák telefonszáma
        - 5;Ben Johnson;+36209874654
    - Felhasználó a programon keresztül tud új adatsort írni és változtatni egy adatsorban lévő telefonszámon

4. Felhasznált beépített könyvtárak

- stdio.h
- stdlib.h
- string.h

5. A modulok és függvényeik

A főmodul (main.c)

- int main()
    - megjeleníti a kezdőképernyőt a mainScreen függvény meghívásával
    - belép egy ciklusba, ami addig fut amíg a mainmenu() függvény visszatérési értéke 1
- int mainmenu()
    - Főmenü megjelenítése
    - továbbítja a felhasználót egy általa választott menübe
    - visszatérési értéke a választásnak megfelelő
- int bookmanagement()
    - belépés előtt megjelenít egy töltőképernyőt a bookLoadigScreen függvény meghívásával
    - Book Management Page menü megjelenítése
    - továbbítja a felhasználót egy általa választott menübe
    - visszatérési értéke a választásnak megfelelő
- int studentmanagement()
    - belépés előtt megjelenít egy töltőképernyőt az stLoadigScreen függvény meghívásával
    - Student Management Page menü megjelenítése
    - továbbítja a felhasználót egy általa választott lehetőséghez, művelethez
    - visszatérési értéke a választásnak megfelelő
- int searchBook()
    - Search for Specific Book Page menü megjelenítése
    - továbbítja a felhasználót egy általa keresési menüpontba
    - visszatérési értéke a választásnak megfelelő

A könyvek adatainak feldolgozására szolgáló, láncolt listát alkalmazó modul (LinkedList.c) és függvényei

- int readOneBookFromFile(bookdata_t **bookFromList, FILE *input)
    - egy darab könyv beolvasását végzi fájlból, melynek azonnal dinamikusan helyet foglal a memóriában
    - a karaktertömb típusok beolvasására meghívja a readStringInputFromFile függvényt, amely beolvassa az adott sztringet és dinamikusan foglal nekik helyet a memóriában
    - visszatérési értéke 1, ha sikeres volt a beolvasás
- void appendBookToFrontOfList(bookdata_t **booklist, bookdata_t *newbook);
    - Egy cím szerint kapott elemet a láncolt lista elejére fűz, vizsgálva, hogy a lista üres-e kezdetben
- bookdata_t *readBooksFromFile (FILE *input);
    - NULL értékre inicializálja a listát
    - meghívja a readOneBookFromFile függvényt addig, amíg az 1-el tér vissza
    - minden beolvasott elemet a lista elejére fűz a ciklusban, az appendBookToFrontOfList függvény meghívásával
    - visszatér a fájlból beolvasott és feltöltött láncolt listával

- void freeBook(bookdata_t *current)
    - egy listaelem felszabadítását végzi
    - előbb felszabadítja a dinamikusan foglalt sztringeket a listaelemben, majd magát a listaelemet is
- void freeAllBooks(bookdata_t *booklist)
    - egy while ciklusban a minden lefutáskor felszabadítja egy elem tartalmát majd magát az elemet, ezzel a végén felszabadul az összes listaelem
- void printList(bookdata_t *booklist)
    - kiíratja a képernyőre a lista elemeit
- char *readStringInputFromFile(FILE *input)
    - fájlból beolvas egy sztringet
    - dinamikusan foglal neki helyet a memóriában
    - visszatér a beolvasott sztringgel
- void printStudentListIntoFile(studentdata_t *studentlist, FILE *input)
    - a paraméterére kapott láncolt listát beírja a Student_Records.txt fájlba, elemenként külön sorokba

A diákok adatainak feldolgozására szolgáló, láncolt listát alkalmazó modul (LinkedList.c) és függvényei

- int readOneStudentFromFile(studentdata_t **StudentOfList, FILE *input)
    - egy darab diák adatainak beolvasását végzi fájlból, melynek azonnal dinamikusan helyet foglal a memóriában
    - a karaktertömb típusok beolvasására meghívja a readStringInputFromFile függvényt, amely beolvassa az adott sztringet és dinamikusan foglal nekik helyet a memóriában
    - visszatérési értéke 1, ha sikeres volt a beolvasás

- void appendStudentToFrontOfList(studentdata_t **studentlist, studentdata_t *current)
    - Egy cím szerint kapott elemet a láncolt lista elejére fűz, vizsgálva, hogy a lista üres-e kezdetben

- studentdata_t *readStudentsFromFile(FILE *input)
    - NULL értékre inicializálja a listát
    - meghívja a readOneStudentFromFile függvényt addig, amíg az 1-el tér vissza
    - minden beolvasott elemet a lista elejére fűz a ciklusban, az appendStudentToFrontOfList függvény meghívásával
    - visszatér a fájlból beolvasott és feltöltött láncolt listával

- void freeStudent(studentdata_t *studentlist)
    - egy listaelem felszabadítását végzi
    - előbb felszabadítja a dinamikusan foglalt sztringeket a listaelemben, majd magát a listaelemet is

- void freeAllStudents(studentdata_t *studentlist)
    - egy while ciklusban a minden lefutáskor felszabadítja egy elem tartalmát majd magát az elemet, ezzel a végén felszabadul az összes listaelem

- void printStudentList(studentdata_t *studentlist)
    - kiíratja a képernyőre a lista elemeit

- char *readStringFromFile(FILE *input)
    - fájlból beolvas egy sztringet
    - dinamikusan foglal neki helyet a memóriában
    - visszatér a beolvasott sztringgel

- void printStudentListIntoFile(studentdata_t *studentlist, FILE *input)
    - a paraméterére kapott láncolt listát beírja a Student_Records.txt fájlba, elemenként külön sorokba

A program alapvető működését segítő belső függvényeket tartalmazó modul (CoreFunctions.c)

- char *readStringFromStdin(void)
    - beolvas egy kapott sztringet a strandard inputrol (billentyűzet)
    - dinamikusan lefoglalja a helyet neki
    - visszatér a beolvasott sztringgel

- int bookIdCheck(int givenID)
    - megnyitja a Book_Records.txt fájlt, beolvassa a láncolt listába és ellenőrzi, hogy a paraméterként kapott azonosító szám tartozik-e már egy másik könyvhöz
    - ha nem, akkor 0-val, ha igen akkor 1-gyel, hiba esetén -1-gyel tér vissza
    - használat után fel is szabadítja a listát

- int studentIdCheck(int givenID)
    - megnyitja a Student_Records.txt fájlt, beolvassa a láncolt listába és ellenőrzi, hogy a paraméterként kapott azonosító szám tartozik-e már egy másik diákhoz
    - ha nem, akkor 0-val, ha igen akkor 1-gyel, hiba esetén -1-gyel tér vissza
    - használat után fel is szabadítja a listát

- int studentExists(int givenid)
    - megnyitja a Student_Records.txt fájlt, beolvassa a láncolt listába és ellenőrzi, hogy a paraméterként kapott azonosító szám létezik-e (van-e ilyen azonosítójú diák)
    - ha létezik, akkor 0-val, ha nem akkor 1-gyel, hiba esetén -1-gyel tér vissza
    - használat után fel is szabadítja a listát

- void printStudent(int studentid)
    - megnyitja a Student_Records.txt fájlt, beolvassa a láncolt listába majd kiírja a képernyőre a paraméterként kapott azonosítóhoz tartozó diák nevét
    - használat után fel is szabadítja a listát

- int deleteContentsOfBookFile()
    - megnyitja Book_Records.txt fájlt és kitörli a tartalmát
    - ha sikeres volt a törlés, akkor 0-val tér vissza

- void saveBookDataIntoFile(bookdata_t *listOfBooks)
    - meghívja a deleteContentsOfBookFile() függvényt, ha az 0-val tér vissza akkor megnyitja a Book_Records.txt fájlt
    - Ezután a printListIntoFile függvény meghívásával beírja a fájlba a paraméterére kapott láncolt listát

- int deleteContentsOfStudentFile()
    - megnyitja Student_Records.txt fájlt és kitörli a tartalmát
    - ha sikeres volt a törlés, akkor 0-val tér vissza

- void saveStudentDataIntoFile(studentdata_t *listOfStudents)
    - meghívja a deleteContentsOfStudentFile() függvényt, ha az 0-val tér vissza akkor megnyitja a Student_Records.txt fájlt
    - Ezután a printStudentListIntoFile függvény meghívásával beírja a fájlba a paraméterére kapott láncolt listát

- void printOneBook(bookdata_t *current)
    - kiírja a képernyőre a paraméterére kapott listaelem tartalmát

- void credits()
    - kirajzol egy „elköszönő” képernyőt

- void stLoadindScreen()
    - kirajzol egy töltőképernyőt a Student Management Page-hez

- void bookLoadindScreen()
    - kirajzol egy töltőképernyőt a Book Management Page-hez

- void searchLoadingScreen()
    - kirajzol egy töltőképernyőt a Search for Book Page-hez

- void mainScreen()
    - kirajzolja a kezdőképernyőt

A Book Management Page-hez tartozó műveletek függvényei (BookManagementFunctions.c)

- int addBook()
    - új könyv hozzáadását végzi
    - ellenőrzi, hogy a felhasználó által megadott azonosító megfelelő-e (nagyobb, mint nulla)
    - a bookIdCheck függvény meghívásával ellenőrzi, hogy egyik másik könyvnek semaz az azonosítója
    - addig kér be újat, amíg az előző két feltétel nem teljesül
    - ellenőrzi, hogy a felhasználó által megadott kiadási év megfelelő-e (0 és 2022 közötti szám), addig kér be újat amig a feltétel nem teljesül
    - felhasználó által megadott adatokat a fájl végére írja
    - a karaktertömb típusú adatoknak dinamikusan foglal memóriát, amit a művelet elvégeztével felszabadít
    - visszatérési értéke 2

- int bookAvailability()
    - megnézi a könyvek listájában, hogy egy adott könyv hol van
        - ha a könyvtárban van (adat 0) akkor kiírja, hogy a könyvtárban van
        - ha egy diáknál, akkor kiírja a diák azonosítóját

- int listBook()
    - beolvassa a listába fájlból a könyvek adatait a readBooksFromFile függvény meghívásával
    - a printList függvény meghívásával kilistázza őket a képernyőre
    - a freeAllBooks függvény meghívásával felszabadítja a listát
    - visszatérési értéke 2

- int deleteBook()
    - bekér a felhasználótól egy azonosítót
    - betölti a Book_Records.txt fájl adatait egy láncolt listába
    - megkeresi a bekért azonosítójú könyvet (listaelemet), majd kifűzi azt a listából, vizsgálva, hogy a honnan kell kifűzni (lista eleje, közepe, vége)
    - kifűzés után felszabadítja az elemet
    - ezután a saveBookDataIntoFile függvény segítségével elmenti a megmaradt listát fájlba
    - visszatérés előtt felszabadítja a teljes listát
    - visszatérési értéke 2

- int modifyBook()
    - bekér a felhasználótól egy azonosítót
    - betölti a Book_Records.txt fájl adatait egy láncolt listába
    - megkeresi a bekért azonosítójú könyvet (listaelemet), majd bekéri a könyv új adatait
    - kiírja a változtatás előtti és utáni adatsort
    - ezután a saveBookDataIntoFile függvény segítségével elmenti a változtatott listát fájlba
    - visszatérés előtt felszabadítja a teljes listát
    - visszatérési értéke 2

- int rentBook()
    - bekér a felhasználótól egy azonosítót
    - vizsgálja, hogy létezik-e ilyen azonosítójú könyv
    - vizsgálja, hogy ki van-e adva az a könyv
    - betölti a Book_Records.txt fájl adatait egy láncolt listába
    - megkeresi a bekért azonosítójú könyvet (listaelemet), majd bekéri a diák azonosítóját, akinek ki lesz adva a könyv
    - vizsgálja, hogy létezik-e ilyen azonosítójú diák
    - ezután a saveBookDataIntoFile függvény segítségével elmenti a változtatott listát fájlba
    - visszatérés előtt felszabadítja a teljes listát
    - visszatérési értéke 2

- int listRentals()
    - betölti a Book_Records.txt fájl adatait egy láncolt listába
    - végigfut a listán
    - ahol a „where” adat nem 0 (tehát ki van adva egy diáknak), ott kiírja a könyv nevét és a printStudent függvény meghívásával kiírja a diák nevét is, aki kikölcsönözte

A Search for Book menü műveleteinek függvényei (SearchFunctions.c)

- int searchbyID()
    - betölti a Book_Records.txt fájlt egy láncolt listába
    - bekéri a felhasználótól a keresett könyv azonosítóját
    - a felhasználó által megadott azonosítójú könyvet megkeresi a könyvek listájában és kiírja a könyv adatait
    - ha nincs ilyen könyv akkor ezt írja ki
    - felszabadítja a listát a freeAllBooks függvény meghívásával
    - visszatérési értéke 4

- int searchbyTitle()
    - betölti a Book_Records.txt fájlt egy láncolt listába
    - bekéri a felhasználótól a keresett könyv címét
    - a felhasználó által megadott című könyv(ek)et megkeresi a könyvek listájában és kiírja a könyv(ek) adatait
    - ha nincs ilyen könyv akkor ezt írja ki
    - felszabadítja a listát a freeAllBooks függvény meghívásával
    - visszatérési értéke 4

- int searchbyAuthor()
    - betölti a Book_Records.txt fájlt egy láncolt listába
    - bekéri a felhasználótól a keresett könyv íróját
    - a felhasználó által megadott író könyvét/könyveit megkeresi a könyvek listájában és kiírja a könyv(ek) adatait
    - ha nincs ilyen könyv akkor ezt írja ki
    - felszabadítja a listát a freeAllBooks függvény meghívásával
    - visszatérési értéke 4

- int searchbyYear()
    - betölti a Book_Records.txt fájlt egy láncolt listába
    - bekéri a felhasználótól a keresett könyv kiadási évét
    - a felhasználó által megadott évben kiadott könyv(ek)et megkeresi a könyvek listájában és kiírja a könyv(ek) adatait
    - ha nincs ilyen könyv akkor ezt írja ki
    - felszabadítja a listát a freeAllBooks függvény meghívásával
    - visszatérési értéke 4

- int searchbyCategory()
    - betölti a Book_Records.txt fájlt egy láncolt listába
    - bekéri a felhasználótól a keresett könyv kategóriáját (témáját)
    - a felhasználó által megadott kategóriába tartozó könyv(ek)et megkeresi a könyvek listájában és kiírja a könyv(ek) adatait
    - ha nincs ilyen könyv akkor ezt írja ki
    - felszabadítja a listát a freeAllBooks függvény meghívásával
    - visszatérési értéke 4

A Student Management Page-hez tartozó műveletek függvényei (BookManagementFunctions.c)

- int addStudent();
    - új diák hozzáadását végzi
    - megynitja a Student_Records.txt fájlt
    - ellenőrzi, hogy a felhasználó által megadott azonosító megfelelő-e (nagyobb, mint nulla)
    - a bookIdCheck függvény meghívásával ellenőrzi, hogy egyik másik könyvnek sem ez az azonosítója
    - addig kér be újat, amíg az előző két feltétel nem teljesül
    - ellenőrzi, hogy a felhasználó által megadott kiadási év megfelelő-e (0 és 2022 közötti szám), addig kér be újat amig a feltétel nem teljesül
    - felhasználó által megadott adatokat a fájl végére írja
    - a karaktertömb típusú adatoknak dinamikusan foglal memóriát, amit a művelet elvégeztével felszabadít
    - visszatérési értéke 3

- int modifyStudent()
    - bekér a felhasználótól egy azonosítót
    - betölti a Student_Records.txt fájl adatait egy láncolt listába
    - megkeresi a bekért azonosítójú diákot (listaelemet), majd bekéri a diák új telefonszámát, a régi adatot erre módosítja
    - kiírja a változtatás előtti és utáni adatsort
    - ezután a saveStudentDataIntoFile függvény segítségével elmenti a változtatott listát fájlba
    - visszatérés előtt felszabadítja a teljes listát
    - visszatérési értéke 3

- int rentedbystudent()
    - bekér a felhasználótól egy azonosítót
    - a studentExists függvény hívásával megnézi, hogy létezik-e ilyen azonosítójú diák
    - ha igen, akkor betölti a Book_Records.txt fájl adatait egy láncolt listába
    - megkeresi a diákhoz tartozó könyvet vagy könyveket és azokat kiírja a képernyőre
    - ha nem vett ki könyvet, akkor ezt írja ki
    - a művelet végén felszabadítja a listát
    - visszatérési értéke 3