Programozói dokumentáció-Prog1 nyilvántartás-Félkész

HUBX6D-Sági Barnabás

Szükséges fájlok:

- DIAK.txt: Ebben vannak eltárolva a hallgatók adatai, a formátuma pedig minden esetben egy vesszőkkel elválasztott szöveg, amely hallgatónként sorokra van osztva, és a benne szereplő adatok ebben a sorrendben a következőek:
    - Név: tetszőleges hosszú, de maximum 50karakter(+ a lezáró '\0'->ez mindenhol ott van, többihez nem írom ki).
    - Neptun-kód: pontosan 6 karakter.
    - KisZH: pontosan 2 karakter(szám).
    - ZH: pontosan 2 karakter(szám).
    - Házifeladat: pontosan 2 karakter(szám).
    - Labor száma: pontosan 2 karakter(szám).
    - Labor neve: pontosan 3 karakter.
    - Gyakorlat neve: pontosan 3 karakter.
    - Előadás neve: pontosan 2 karakter.
- OKTATO.txt: Ebben vannak eltárolva az oktatók adatai, megegyezik a DIAK.txt formátumával, annyi kivétellel, hogy csak a nevet, labor nevet, gyakorlat nevet és előadás nevet tárolja.
- GYAK.txt: Itt soronként egy darab gyakorlat név kerül tárolásra.
- LAB.txt: Itt soronként egy darab labor név kerül tárolásra.

A program:

- **main.c:** Ez az alapköve, innen léphetünk be a megfelelő menübe és léphetünk ki a programból. A megfelelő menü kiválasztására a switch-esetszétválasztásos módszer tűnt a leghatékonyabbnak. A programból való kilépést egy do-while hátultesztelős ciklus segíti, mely a megfelelő adatra('9') befejezi a futást, ezzel vége is a programnak.
- **Második lépcsős menük(gyakeslabmenu.c, oktatokmenu.c, hallgatokmenu.c):** Működésben megegyeznek a main-nel, a menpontjaik a kilépések kivételével egy-egy függvényt hívnak meg. Egyetlen függvény hajtódik végre, melyek a main függvénynek csak egy bool értéket adnak vissza.
- **structok.h:** Itt vannak a megfelelő struktúrák(minden egyes adattípushoz), melyek igazodnak a fent említett formátumokhoz, illetve felépítésükben megfelelnek, hogy láncolt listát lehessen velük felépíteni.
- **beolvas.c:** Itt struktúránként szétválasztva(külön függvénnyel), soronként beolvasásra kerülnek a txt fájlok. Minden egyes új elemmel létrejön egy láncolt lista következő eleme, melybe egyből betöltésre kerülnek a megfelelő adatok. A módszer, amellyel ezt a betöltést végzi a program elég átlátható, a nevet az első ',' karakterig olvassa be, a

többi adatot pedig(mivel azoknak a hossza mindig megegyezik) könnyen beolvashatóak egy kis matekkal. A beolvasás során a láncolt lista(név szerint, ABC rendben) rendezett formában épül fel, ezzel nagyban megkönnyítve a programban futó sorba rendezéseket. Két féle felépítést valósítottam meg, az egyik a nevek szerinti, másik a pontok szerinti(ez szükséges a pontok szerinti kiíráshoz). Ennek állítását egy egyszerű bool visszatérési értékkel rendelkező segédfüggvény végzi, mely az állítandó értéktől függően kapja az igen-nem instrukciót, amely már szükséges a beolvas_diak függvény meghívásakor is. A beolvasással foglalkozó függvényeknek a beolvas_diakon kívül nincs paraméterük, visszatérési értékük pedig a lista első eleme. A függvények tudják kezelni a nem megfelelő fájlmegnyílást, ekkor NULL-al térnek vissza.

- **kiir.c:** Struktúránként szétválasztva(külön függvényekkel) fájlba írja az adatokat, a megfelelő formátumban(láncolt listán megy végig). A függvényeknek nincs visszatérési értéke, paraméterük pedig a lista első eleme. A rendezett beolvasás miatt, a kimenetre már a rendezett láncolt lista kerül kiírásra(csak a névsor szerint rendezett, mert a másik sehol nem kerül kiírásra.) A függvények tudják kezelni a nem megfelelő fájlmegnyílást, illetve azt is ha elem nélküli listát kapnak(for el sem indul).

- **felszabadit.c:** Struktúránként egyszerű felszabadító függvények, melyek láncolt listákra szakosodtak. Nincs visszatérési értékük, paraméterük a lista első eleme, tud kezelni elem nélküli listát is(NULL).

- **keresesek.c:** Itt megvalósulnak a megfelelő keresések diák név, diák neptun-kód, oktató név alapján, egyszerű keresés láncolt listában, melyet a beolvasás megfelelő függvényével csinál, majd a megfelelő adatok kiírása, ezek után a beolvasás során lefoglalt memória felszabadítása a megfelelő felszabadító függvénnyel. A függvények nem adnak vissza értéket és hívásuk során sincs szükségük paraméterekre, egyszerűen kiírják a láncolt listát, illetve visszajeleznek a nincsilyen segédfüggvény(listaz.c-ben van) segítségével, ha nincs találat. Képesek kezelni az elem nélküli(NULL, a for „ellenőrzi”)listákat.

- **hozzaad.c:** Itt történik meg az új oktató, hallgató, gyakorlat, labor felvétel. A függvények kissé hosszúak, de a kérdések miatt nem igazán lehet rövidebbeket készíteni. A kérdések után kerülnek beolvasásra a megfelelő adatok. A függvények először lefoglalnak egy új láncolt lista elemet, majd a beolvasással kapcsolatos függvényt meghívják, ekkor létrejön a meglévő adatokból egy láncolt lista és ehhez fűzik az új elemet. Ezek után kiírják a megfelelő függvény segítségével a fájlba az új láncolt listát(mely egy elemmel bővül), majd felszabadítják a lefoglalt memóriát. A függvények nem adnak vissza értéket és hívásuk során sincs szükségük paraméterekre. Képesek kezelni az eddig elem nélküli(NULL, a while „ellenőrzi”)listákat, ebben az esetben egy elemű listában létrehozzák az első elemet.

- **listaz.c:** A függvények a megfelelő kritériumok alapján kilistázzak a megfelelő adathalmazokat. Itt van pár „helyi”(static), illetve sima segédfüggvényem, amik segítenek a jobb átláthatóságban, hiszen ez a .c egy elég hosszúra sikeredett programrész lett. A függvények minden esetben beolvasnak, kiírnak(kimenetre, nem fájlba), majd a beolvasás által lefoglalt memóriát felszabadítják. A függvények nem adnak vissza értéket és hívásuk során sincs szükségük paraméterekre. Az osztalyzat segédfüggvény az osztályzatot számolja ki a pontok összege alapján(paraméter), majd visszatér a jeggyel. A nincsilyen segédfüggvény meghívásra kerül a keresések során, amennyiben hamis érték a paramétere, vagyis a függvény amely meghívta nem talált senkit, akkor ezt ki is írja(visszajelzés). A kovetelmenyek segédfüggvény egyszerűen

a négy paraméter alapján ellenőrzi a követelmények meglétét, visszatérési értéke pedig egy bool válasz. Képesek kezelni az elem nélküli(NULL, a for „ellenőrzi”)listákat.

- **modosit.c:** Működésben a keresés és a hozzáadás szerelemgyereke, de programszinten nincs közük egymáshoz, szerintem így tisztábban követhetőek. A függvények a megszokott beolvasás után, egy egyszerű láncolt listán végigmenetel közben megkeresik a szükséges adatot, amennyiben ez megvan, akkor lehet a módosítható adatokat újraírni. A módosított adatok, vagyis módosított láncoltlista kiírásra kerül a fájlba, majd felszabadul a memória a megfelelő függvénnyel. A függvények nem adnak vissza értéket és hívásuk során sincs szükségük paraméterekre, ellenben ők többször hívnak meg más függvényeket, ezen kívül minden esetben visszajelzést adnak a műveletek sikerességéről/sikertelenségéről. Képesek kezelni az elem nélküli(NULL, a while „ellenőrzi”)listákat.
- **torol.c:** Hasonló a kereséshez, de itt sincs összekötve programszinten. Beolvasás után a láncolt listán végigmenve, a megfelelő adat alapján megkeresésre kerül a törlendő elem, amely törlésre kerül a láncolt listából(akkor is működik, ha első, utolsó, illetve ha egyik sem a keresett elem). Ezek után egyből fájlba íródik az új láncolt lista és felszabadítódik a memória. A függvények nem adnak vissza értéket és hívásuk során sincs szükségük paraméterekre, ellenben ők többször hívnak meg más függvényeket, ezen kívül minden esetben visszajelzést adnak a műveletek sikerességéről/sikertelenségéről. Képesek kezelni az elem nélküli(NULL, a while „ellenőrzi”)listákat, akár elem nélküli listával is meghívhatják a kiírós függvényt(ez ott van kezelve).

- **Egyéb:**
    - **debugmalloc.h**: Minden forrásfájlban szerepel, segítségével leellenőrizhetőek a memóriával kapcsolatos műveletek. Az ellenőrzés során minden működik, a memóriaműveltek tökéletesek.
    - **fejlécfájlok(.h):** Értelemszerűen működnek, minden forrásfájlnak megvan a sajátja. Minden forrásfájlban a megfelelő fejlécfájl került beillesztésre(ezt abból is lehet tudni, hogy nem dobál fel állandóan warningokat).
    - **string.h, stdbool.h könyvtárak:** Abban az esetben kerültek beillesztésre, amikor szükség volt rájuk(bool esetén->ezek általában a hibaüzenetek miatt kerültek beépítésre, illetve sztringfüggvények esetén(strcmp), mely az összes kereséssel kapcsolatos függvénybe be van építve).