Névjegyzék program fejlesztői dokumentáció

A program általános jellege

Egy névjegyzék mindenkinek jól jön, amikor el kell mentenie az ismerőseinek adatait. Ez a program éppen erre alkalmas. Tárolja az adatokat egy adatbázisban. A felhasználónak lehetősége van hozzáadni új névjegyet; keresni a névjegyek között majd a kilistázott névjegyekből kiválasztani egyet és azt módosítani vagy törölni. Név alapján való keresés esetén van lehetőség egy wildcard karakter (*) használatára is. (pl. a "Nagy*", "N*án", "*István" keresősztringek mindegyike megtalálja Nagy Istvánt a névjegyzékben)

Emellett még arra is van lehetőség, hogy vCard formátumú fájlból importáljon az adatbázisba, valamint egy kiválasztott névjegy vCard formátumú exportálására is van mód.

Minden, az adatbázis változtatásával járó műveletet követően automatikusan mentés kerül az adatbázis fájlba is, erre a felhasználónak nem kell figyelnie a programból való kilépés előtt.

Az adatszerkezet

A program névjegyekkel dolgozik, így egy Névjegy struktúra felvétele egy ideális kezdés.

```
typedef struct Nevjegy{
   int id;
   char nev[30+1];
   char foglalkozas[20+1];
   char cim[30+1];
   char telefonszam[12+1];
   char email[20+1];
   struct Nevjegy *elozo, *kov;
} Nevjegy;
```

A fentebb látható módon minden adatmezőnek fix hossza van. Ezek alapján dolgozik a program a névjegyekkel. Ideiglenes névjegy tárolókból több is szerepel a programban, azonban az adatbázis adatszerkezete egy duplán láncolt lista. Ezt az indokolja, hogy egy névjegy törlésénél fontos, hogy ne kelljen az egész adatbázis számára újra memóriát foglalni. Emellett negatívuma nem igazán van, mivel a program alapját adó keresés műveletnél mindenképpen végig kell menni az adatbázison, indexelhetőség nem segítene.

Egy névjegy hozzáadásakor kap egy id-t, ami egyedi az adatbázisban. Ezzel lehet majd azonosítani a keresés funkció kiválasztás részénél.

Modulok

A program a main forrásfájlon kívül három modulból épül fel: adatbázis, fájlkezelés, input/output (a felhasználóval való kommunikáció). Ezek rendre az adatbázis kezelésével kapcsolatos, a fájlkezelésért felelős, és a felhasználóval való kommunikációt segítő függvényeket tartalmazzák; de lehetnek kis átfedések is.

A program működését vezérlő fő függvények

A felhasználó elsősorban a menü által kommunikál a programmal.

void menu(Nevjegyzek* adatbazis);

Ez a függvény folyamatosan kiír egy menüt a kijelzőre, amikor nem más funkció kiírásait kell megjeleníteni. Az adatbázist továbbítania kell az alatta levő függvények felé, így annak helyét paraméterként megkapja. Működési elve, hogy addig kér bemenetet, amíg a kilépést nem választja a felhasználó. Felhasználói hiba a default esetben van kezelve. Ilyenkor nem csinál semmit.

void hozzaadas(Nevjegyzek* adatbazis);

Ez a függvény a névjegy hozzáadására szolgál, eszerint kezeli az alatta levő függvényeket. Paraméterként ő is megkapja az adatbázis helyét, mivel majd tovább kell adnia a többi függvénynek. Működési elve: először bekéri az adatokat egy ideiglenes tárolóba, majd, ha a felhasználó is megerősíti, akkor elmenti azokat az adatbázisba.

void kereses(Nevjegyzek* adatbazis);

Ez a függvény a névjegyzékben való keresésre szolgál. A felhasználótól kapott kategória szerint keres az adatbázisban, és kilistázza a megfelelő elemeket. Ezután van lehetőség kiválasztani egyet id alapján a névjegyek közül, majd ezt a névjegyet kiíratni, módosítani vagy törölni.

void importalas(Nevjegyzek* adatbazis);

Ez a függvény a vCard formátumú fájlok importálására való. Ha a megadott fájlnév végéről hiányzik a kiterjesztés, azt a hibát kijavítja, és működni fog.

Ezeket a függvényeket a main.c fájl tartalmazza.

Adatbázis kezelő fő függvények

bool adatb_letrehoz(Nevjegyzek* adatbazis);

Létrehozza az adatbázist, és ha van már adatbázis fájl, akkor betölti annak tartalmát. Ha valami nem sikerült, akkor hamis értékkel tér vissza, egyébként igazat. Ha bármi nem sikerül a memória foglalásból vagy a fájlmegnyitásból, hamis értéket ad vissza, és ez kezelve lesz a main-ben. (Fel lesz szabadítva minden lefoglalt memória terület, majd kilép a program.)

void mentes(Nevjegy const *const nevjegy, Nevjegyzek* adatbazis);

Ez a függvény egy ideiglenes névjegy tárolóból átmenti az adatokat az adatbázisba, illetve az adatbázis fájlba. Ezekre függvényeket hív meg. Az új névjegy id-jának megadásához is függvényt hív meg.

bool memoriaba_iras(Nevjegy const* const nevjegy, Nevjegyzek* adatbazis);

Egy ideiglenes névjegy tárolóból átmenti az adatokat az adatbázis által foglalt memóriaterületre. Ez a függvény végzi a memóriafoglalást. A felszabadítás a felszabadító függvény hívásakor megtörténik.

void nevjegy_id_general(Nevjegyzek const * const adatbazis, Nevjegy* const nevjegy);

A paraméterként kapott névjegynek generál egy id-t és el is menti bele. (Elég lenne a legutolsó névjegy id-ját megnézni, mivel id szerint sorba van rendezve mindig az adatbázis, de így esetleges változtatásokhoz is jobban tud alkalmazkodni a program)

bool nevjegy_torles(Nevjegyzek const * const adatbazis, Nevjegy * const torlendo)

Ez a függvény végzi a névjegy törlését. Törléskor az adatbázis fájl teljesen újra íródik. Hibalehetőség lehet, ha nem sikerül megnyitni a fájlt.

bool nevjegy_modosit(Nevjegyzek* const adatbazis, Nevjegy const * const nevjegy, Nevjegy* const eredeti)

Ez a függvényt az adatbázisban szereplő névjegy módosítását végzi, majd újra írja az adatbázis fájlt. Akkor ad vissza hamis értéket, ha nem sikerül megnyit a fájlt.

void adat_fixalas(Nevjegy* const nevjegy);

Ez a függvény csak az adatbevitelkor üresen hagyott mezőket tölti ki n/a-val. Azért fontos, mivel az adatbeolvasás nem működne, ha üres mező íródna az adatbázis fájlba.

bool wildcard_keres(char const * const kereso_string, char const * const szoveg)

Visszaadja, hogy megfelel-e a paraméterként kapott kereső stringnek a másik paraméterként kapott szöveg. Ez a függvény egy regex függvény leegyszerűsített változata. Megírásának oka, hogy ne kelljen külső könyvtárat használni, illetve ez a függvény sokkal egyszerűbb és elég erre a célra.

Három eset történhet benne. Ha az első karakter a *, akkor azt elhagyja; ha az utolsó, akkor onnan hagyja el; ha valahol a közepén, akkor szétszedi a kereső stringet és egyenként megnézi, hogy megfelel-e a szövegnek. Ha az első rész nem felel meg, akkor a másodikat már nem ellenőrzi. Ha az első megfelelt, akkor azt is vizsgálja, hogy a második rész az első után helyezkedik-e el.

Fájlkezelő fő függvények

Ha nem sikerült megnyitni az adott fájlt, általában akkor adnak hamis értéket vissza.

bool van_adatb_fajl(void);

Megnézi, hogy létezik-e már adatbázis fájl.

bool adatb_beolvas(Nevjegyzek* adatbazis);

Beolvassa az adatbázis fájlból az adatokat az adatbázis által foglalt memóriahelyre. A fájl fejlécét átugorva soronként szerzi meg az adatokat. Teszteléskor kiír minden beolvasott névjegyet.

bool fajl_ inicializal_adatb(void);

Létrehozza vagy felülírja az adatbázis fájlt. Beleírja a fejlécet.

bool fajlba_adatbazis_mentes(Nevjegyzek const * const adatbazis);

Ez a függvény írja ki az adatbázist a fájlba. Újraírja a fájlt.

bool vcard_import(char* fajlnev, Nevjegyzek* adatbazis);

A paraméterként kapott fájlnevű vCard fájlban szereplő névjegyet importálja az adatbázisba. Ha nincs megadva fájlkiterjesztés, akkor hozzáírja. Soronként beolvassa a fájlt, majd megnézi, hogy megfelel-e valamelyik szükséges adatmezőnek. Ha egy adatmező nem lett kitöltve, akkor az ***adat_fixalas*** függvény azt kijavítja. Csak ASCII kódolású fájlból tud beolvasni helyesen.

bool vcard_export(Nevjegy const * const nevjegy);

A névjegyben szereplő nevet használva fájlnévként exportálja a névjegyet ASCII kódolású 4.0 verziójú vCard fájlba.

A felhasználóval való kommunikációra szolgáló fő függvények

void adatbevitel(Nevjegy* nevjegy, bool* mentes_e);

Egy személy adatainak egyesével való bevitelére szolgál. Ha már létező névjegyet kap, akkor kiírja a benne szereplő adatokat. Végül meghívja az ***adat_fixalas*** függvényt.

void nevek_listaz(Nevjegyzek const *const adatbazis);

A választott kategória alapján keres és kilistázza a megfelelő neveket id-val együtt.

Általános gondolatok

A program nem tartalmaz semmi nem triviális algoritmust. Csak beépített könyvtárakra van szükség a fordításához (stdio.h, string.h, stdbool.h, stdlib.h, windows.h), illetve a debugmalloc.h fájlra. Ennek megtartása érdekében saját kereső függvényt használ, nem pcre-beli regex függvényeket.

Általában a visszalépés a v karakter beírásával lehetséges, bár sok helyen bármilyen más karakter esetén is csak visszalépés történik. Így nem lehet kiakasztani a programot egyéb karakterek megadásával.

Az adatok bevitelénél nincs ellenőrzés arra, hogy milyen karaktereket ír be a felhasználó, mivel nem olyan fontos a program szempontjából, hogy például telefonszámhoz ne szöveget írjon be. Így általánosabban, megkötések nélkül használható a program.

A program csak Windows operációs rendszerre lett tervezve, máshol nem volt tesztelve sem.

Az importálás és exportálás csak ASCII kódolású fájllal működik rendesen. UTF-8 kódolás esetén például az ékezetes karakterek nem fognak megfelelően importálódni.