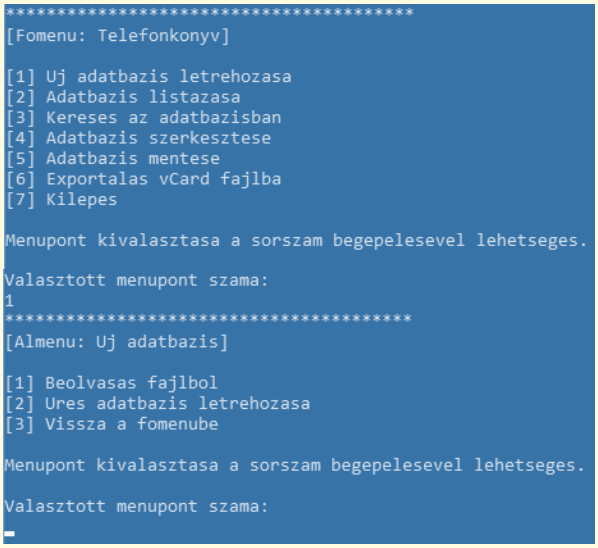
NHF:Telefonköny

Nyilatkozat a program jelenlegi állapotáról, funkcióiról

A program jelenleg futtatható állapotban van, a specifikált funkciók közül több is a végleges állapothoz közelít. Azt tapasztaltam a megvalósítás során, hogy jelentősen több időre van szükség az egyes függvények megírásához, mint amennyire előre gondoltam. Ennek megfelelően a program folyamatosan fejlődik, minden nap kerülnek bele javítások, illetve próbálom a kezdetleges, kevésbé optimális megoldásokat is megtalálni, miközben a további funkcionalitás eléréséért is dolgozom. A program célját a specifikációban részletesen ismertettem, röviden annyit róla, hogy egy fix 20 mezős adatbázist lesz képes kezelni, és annak a tartalmát vcf formátumú fájlba kiírni/abból beolvasni.



Ezek után szeretnék a konkrét megvalósításról is írni. Ez a szöveg csak a program jelenleg működő funkcióit feltételezi. A program konzolos felületen működik, a feladatok közötti navigálásról karaktervezérelt főmenü, és több kisebb almenü gondoskodik. A tárolt adatok együttesére szeretnék mostantól „adatbázisként” hivatkozni. Ezt láncolt listás adatszerkezetben valósítottam meg. A program egy globális logikai változóban tárolja el, hogy van-e éppen aktív adatbázis, ez induláskor hamis. Aktív adatbázist kétféle módon kaphatunk: úgynevezett „üres adatbázis” létrehozásával, ami egy rekordnyi helyet foglal a memóriában, az ezután kézileg megadott rekordokat ez elé fűzi, így a lista címe mindig az utoljára bevitt elem helye. A másik beviteli mód a fájlból beolvasás. Ezt a funkciót valójában azért terveztem bele a programba, hogy a menüből elérhető „mentés” opció segítségével generált fájlt meg lehessen nyitni, így két használati alkalom között a kezelt adatbázis megmarad(hat). Ehhez a CSV fájlformátumot választottam, mivel ez könnyen kezelhető, és tábálázatkezelővel megnyitva vizuálisan is könnyű kiigazodni rajta. További előny, hogy az így összerakott táblázatokat a programba töltve a lehető leggyorsabb az adatbevitel. A program ugyan rendelkezik rekordfeltöltő függvénnyel, de ez egyelőre még kiforratlan, használata esetleg kényelmetlen lehet. Az újonnan lefoglalt rekordot mezőit a program alapból „nincs” szövegre inicializálja. A beolvasott adatok feldolgozása hibátlanul működik, a feldolgozás után mentett kimenet tökéletesen megegyezik az eredeti bemenettel. Beolvasott adatbázishoz újabb rekordokat adhatunk. A „Listázás” funkció a képernyőre írja az adatbázisban szereplő neveket. A most működő funkciók nagy része képes alapvető hibakezelésre. A forráskódot két állományra bontottam, a jobb átláthatóság érdekében. Így a beküldött állományok halmaza a következő: a **main.c**, ami a főprogram kódját tartalmazza, **adatkezeles.c** és **.h**, a függvények és a hozzájuk tartozó header fájl, valamint egy **pelda.csv** mintafájl, ami egy két példa rekordot tartalmaz.

FÜGGVÉNYLISTA

```
void felszabaditas(Rekord* ab);   //hagyományos algoritmus láncolt lista felszabadítására
Rekord* ures_ab();                  //lefoglal egy rekordnyi helyet, inicializációs lépés
Rekord* uj_rekord(Rekord* ab);        //szintén helyet foglal, majd a paraméteként kapott elem   elé fűzi a azt, létrehozva az új első elemet
void uj_feltolt(Rekord* uj);         //a korábban említett feltöltésre szolgáló függvény, végigmegy az egyes paramétereken, de ezeket nem kötelező kitölteni
void sorbol_feltolt(Rekord* uj, char* sor);         //fájlkezelésnél használt függvény, a beolvasott sorból ki scanneli a vesszővel elválasztott 20 paramétert
Rekord* betoltes(Rekord* ab);                    //CSV fájl beolvasást szervező függvény
void init(Rekord* uj);                                 //rekord mezőinek alapértéket ad
void fajlba_kiir(Rekord* ab);                  //CSV fájl létrehozását szervező függvény
int szerkeszt();                             //almenüt és kiértékelését megvalósító függvény
int uj_adatbazis(Rekord* ab);             //almenüt és kiértékelését megvalósító függvény
void kereses(Rekord* ab);                          //keresés almenü függvény kezdeménye
int menuvalaszt(int max);             //generalizált bemenetellenőrző függvény a menükhöz
void listazas(Rekord* ab);                        //nevek megjelenítése listabejárással
void menulista();                                                 //főmenü megjelenítese
char kitoltes(char* mezo);                 //karakterbemenetet kiértékelő segédfüggvény
void sikertelen(char* hiba);                          //egyszerű egységesített hibakiíró
void pre();                                          //kezdetleges bemenetre váró függvény
bool dbcheck(Rekord* ab);     //adatbázis nem-létezését, valamint ürességét ellenőrző fv.
```

A következő menüpontokat tartalmazza a program:

-Új adatbázis

(a program megkérdezi szeretnénk e menteni a jelenlegit ha van ilyen, akkor fájlnevet kér)

-->Beolvasás fájlból ✓

(a program fájlnevet kér (.csv vagy formátum))

-->Üres adatbázis létrehozása ✓

-Listázás ✓

(megjeleníti az adott adatbázisban eltárolt neveket)

-Keresés ✗

(a program először a keresendő paraméter típusát kérdezi, aztán a keresendő kifejezést,a találat tárolt paramétereit kilistázza. Névkeresésnél a * kiegészítő karakter használatával névrészlet megadása is elegendő. Ilyen esetben a program kilistázza az összes megfelelő találatot, sorszámozva. A kiválasztott név sorszámát megadva kilistázza a tárolt paramétereket.)

-Szerkesztés

-->Új bejegyzés ✓

-->Módosítás ✗

(Keresés funkciót felhasználva megkeresi a kiválasztott személyt, a felhasználótól megkérdezi,hogy melyik paramétert szeretné módosítani, a kiválasztottnak új értéket ad. Ezután megkérdezi újból, hogy mit szeretne módosítani, ameddig az nem lesz a válasz, hogy semmit.)

-->Törlés ✗

(Keresés funkció segítségével megkeres egy bejegyzést, és jóváhagyás után törli azt.)

-Mentés ✓

(fájlnevet kér (.csv formátum))

-Exportálás ✗

(vCard fájlba, fájlnevet kér)

Ez az ábra az eddigi állapot felmérésének segítését szolgálja. (a két sárga X azt jelképezi, hogy azok a függvények a „Keresés” függvény elkészülését igénylik) Ugyan a nagyja még hátravan, de a program saját fájlkezelése és az adatszerkezet gyakorlatilag végleges alakban vannak, ezek után a többi funkciót már csak le kell kódolni.

Tóth Ádám László

Tóth Ádám László