MINI FACABOOK DOKUMENTÁCIÓ

Kalmár Gábor

XCCGBS

A program 4 modulból áll :

Main Program: Ez felel az utasítások kiadásáért, függvények hívásáért, ebben a modulban van a menü.

Loginfv : Ebben a modulban találhatóak meg a regisztrációhoz, bejelentkezéshez szükséges függvények, később részletesebben ismertetve.

Mainfv: Itt vannak a program további függvényei

Messenger: Üzenetküldéssel kapcsolatos függvények helye.

MAIN PROGRAM

A programban található változók: start, egy segédváltozó, melyet a menüben való navigáció elősegítésére hoztam létre. valaszto, egy integer változó, és ez szabályozza a főmenüt a main programban egy switch() segítségével. id egy string változó, ebben tárolom el az aktuálisan bejelentkezett felhasználó nevét, (maximum felhasználói név: 100 karakter), sok függvénynek bemenete. Exit, egy bool alapú változó, mely a program végét szabályozza, ha igaz az értéke, akkor lép ki a programból. Logged_in szintén egy bool változó, ahogy nevéből kiderül, ha az értéke igaz, akkor valaki sikeresen be van jelentkezve.

LOGINFV

A programrészben található függvények:

bool check(char name[]) : Bemeneti értéknek egy stringet kap és megnézi az *adatok.txt*- ben, hogy foglalt-e már ez a profilnév, ha foglalt, false a visszatérési értéke, így a regisztrációs függvény nem enged regisztrálni ezzel a névvel mégegyszer, ha nem foglalt, visszatérési értéke : true. A függvény egy láncolt listába tölti be az adatokat, és stringkezelő függvénnyel hasonlítja össze az adatokat. Struct AdatokLista a mainfv.h – ban van definiálva.

bool registrate(void): A függvény bekér egy felhasználónevet, meg egy jelszót, meghívja a check függvényt, ha visszatérési értéke true, akkor megnyitja a reg.txt – t és beírja a felhasznalónevet és jelszót tabulátorokkal elválasztva, sor vége jel: ~. (jelszó, felhasználónév max 100 karakter). Majd megnyitja az adatok.txt filet, és létrehoz egy új adatsort. (NEV KOR LAKHELY NEM ~). Neven kívül a többi helyre „-”-t rak. amit később a felhasználó tud majd módosítani. Az egyes elemeket tabulátor választja el, sorvége jel: „~”, Függvény visszatérési értéke igaz, ha sikerült az file-ba írás.

bool login(char *id): A függvény bemeneti értéke egy string, amit a sikeres bejelentkezés következtében megváltoztat, a bejelentkezett felhasználó nevére. A függvény beolvas egy felhasználónevet, és egy jelszót, megnyitja a reg.txt- filet, majd egy lancolt listába beolvassa az adatokat onnan, a sikeres azonosítás esetén (jelszó és felhasználónév egyezés esetén) a függvény visszatérési értéke igaz. A láncolt lista elemeit, dinamikusan foglalja és szabadítja fel a függvény.

MAINFV

A programrészben található függvények és feladataik:

AdatokLista* lista (char *name) : A függvény visszatérési értéke egy struktúrára mutató pointer, mely egy láncolt lista. A függvény bemeneti változója egy név, melyet megkeres az adatok.txt fileba, és az adott felhasználó adatait egy láncolt listába rakja. Ha nem talált ilyen nevű felhasználót, akkor a visszatérési érték egy NULL pointer. A láncolt lista dinamikusan foglalt memórián van.

void szemelyes(char *name) : A függvény bemeneti értéke egy string, az aktuálisan bejelentkezett felhasználó neve, ezt odaadja a lista függvénynek, amely visszaad egy láncolt listát az illető adataival. Ezt követően egy switch() segítségével kiírja az információkat, majd felszabadítja a dinamikusan foglalt memóriát.

void linemodify(int tipus,char *name) : A függvény bemeneti értékei egy integer, amely megadja hogy mit módosítson (lakhely, kor, nem), és egy string mely a felhasználó neve, amit odaad a lista függvénynek, és visszakapott láncolt lista megfelelő „láncszemét módosítja”, a switch() használata segítségével beolvasott adattal. Ezután a módosított láncolt listát odaadja a sormodosit függvénynek. Ezután visszatér a linemodify függvényből.

void sormodosit(AdatokLista *eleje, char *name) : A függvény bemenete egy láncolt lista a módosított adatokkal, és a felhasználó neve. Megnyitja az adatok.txt filet majd láncolt listát csinál a soronként beolvasott adatokból, ellenőrzi, hogy egyezik- e a név a felhaszáló nevével, ha nem akkor átmásolja a helyben létrehozott seged.txt – fileba a sor, ha egyezik akkor a módosított sort másolja át a seged.txt-be. Miután végig ment az adatok.txt-n bezárja mind a 2 txt-t, az adatokat kitörli, a seged.txt -t pedig átnevezi adatok.txt-re, így már a módosult adatok vannak benne. Ezután visszatér a függvény.

void idkereso(int start) : A függvény bemeneti értéke egy start nevű integer, mely azt szabja meg mi alapján szeretnénk keresni (név, kor, lakhely, nem). A függvény beolvassa amit szeretnénk keresni, ezt a kereses változóban tárolja el. Megnyitja az adatok.txt filet, és soronként beolvassa egy láncolt listába az adatokat, majd a start és switch() segítségével ellenőrzi, hogy az adott adat egyezik a keresett adattal, ha igen, akkor kiírja az adott illető információit a képernyőre, majd megy tovább, és újabb ember adatait olvassa be, majd ellenőrzi az adatokat stb… A dinamikusan foglalt területeket felszabadítja a függvény.

void follow(char *name) : A függvény bemeneti értéke egy string, mely a bejelentkezett felhasználó neve. Kér egy nevet, hogy kit szeretnél ismerősnek jelölni, ezt a nev változóban tárolja el, ennek az embernek megjeleníti a személyes adatait, ismerőseit a szemelyes függvény segítségével, ha olyanra kerestünk rá, aki már az ismerősünk, azt jelzi a program. Ha még nem ismerősünk akkor megkérdezi, hogy biztos be szeretnénk e jelölni, ha továbbra is beszeretnénk jelölni, akkor lista függvény segítségével alkotott láncolt listát mely a keresett ember adatait tartalmazza módosítjuk, az utolsó „lánc”- ot átírjuk: („felhasznalo neve” tab * tab ~ tab). A csillag jelenti hogy bejelöltük, de még nem igazolt vissza. Majd a sormodosit függvény segítségével módosítja az adatok.txt – filet. majd felszabadítja a foglalt memóriaterületet és visszatér a függvényből.

void confirm(char *name) : A függvény bemeneti értéke egy string, mely a felhasználó neve. A lista függvény segítségével megkapja a láncolt listát melynek első elemére mutató pointer az „eleje”. A lista ezúttal duplán láncolt, hiszen szükség van visszalépésre is abban az esetben ha „*” ot talált a program az adataink között. Ebben az esetben lista előző elemében tárolt string az illető aki bejelölt minket, ezeket a neveket kiírja a program a képernyőre, majd vár egy beolvas egy stringet, annak az embernek a nevével akit vissza szeretnénk igazolni. („barat”). Mégegyszer meg kell erősíteni a visszaigazolást. Ha elfogadtuk, akkor a program végig megy a láncolt listán, majd ahol * ot talál, és az előzőben „barat” ot ott kitörli a csillagot, magyarán a barat->kov pointert a csillag ->köv re irányítja és felszabadítja a * ra mutató pointer. Ezután, sormodosit függvény segítségével módosítja az adatok.txt filet, így már nem lesz ott a *, és az illető sorába hozzárakja a bejelentkezett felhasználó nevét, ezt is sormodosit függvény segítségével, így kölcsönösen látszik a kapcsolat. (sorok végére mindig kirakja a ~ jelet, hiszen beolvasáskor ezt használja viszonyítási pontnak a lista függvény) Ezután visszatér a függvény.

AdatokLista* ismerosok(char *name) : A lista hoz hasonló függvény, bemeneti értéke egy string, az adatok.txt fileból megkeresi az adott illető adatait, és láncolt listát csinál az ismerőseiből. Az ismerősök ismerősei probléma megoldásához kell ez a függvény. A visszatérési érték egy láncolt listára mutató pointer.

void nemismerosfile(char *name,char *barat) : Bemenete kettő db string, egyik a felhasználó neve, másik az egyik ismerőse. Az ismerosok függvény segítségével megalkotja mindkét embernek az ismerőseit alkotó láncolt listát. Megnyitja olvasásra az ismeros.txt filet, az ott lévő tabulátorokkal elválasztott nevekből is láncolt listát csinál. Megnyitja írásra az ismeroseged.txt filet, ebbe először belemásolja az ismeros.txt adatait. Egy for ciklusban egy bool változó segítségével és stringfüggvényekkel kiszűri az olyan embereket akik a felhasznalónak nem ismerősei, de a felhasználó adott barátjának ismerősei, ezek az emberek, ha még nincsenek benne az ismeros.txt fileba (ismétlés elkerülése miatt) akkor beleírja az ismeroseged.txt fileba. Miután végig ment a for cikluson és megnézett minden ismerősét a felhasználó barátjának. akkor bezárja a két megnyitott filet. Felszabadítja a foglalt memóriát, kitörli az ismeros.txt filet, majd az ismeroseged.txt filet átnevezi ismeros.txt – re.

void ismerosokism(char *name) : A függvény bemenete egy string, a bejelentkezett felhasználó neve. Ez a függvény kezeli az előző két függvényt. ismerosok függvény segítségével megalkotja a felhasználó láncol listáját majd, végig megy a listán és minden egyes ismerősét átadja a nemismerosfile függvénynek, majd onnan visszatérve kiiratja az ismeros.txt file tartalmát, amely, csak olyan embereket tartalmaz, akik neki nem az ismerősei de valamely ismerősének igen. Ezután visszatér a függvényből.

Messenger:

A programrészben található függvények és feladataik:

A program modulban van egy másik struktúra az üzenetekhez, (üzenetek maximális hossza : 10000 karakter)

```
typedef struct Uzenetlista{
   char text[10001];
   struct Uzenetlista *kov;
}Uzenetlista;
```

void sendmess(char *uzenet, char *name1, char *name2) : A függvény bemeneti értékei 3 db string, az első az üzenet, a második a feladó neve, a harmadik a címzett neve. A függvény megnyitja a uzenet.txt filet „append” – ra és tabulátorokkal elválasztva fileba írja a először a feladót, majd címzettet, legvégül az üzenetet. Itt is a sor végét ~ karakter és tab zárja, ezután bezárja a filet, és visszatér a függvényből.

void messenger(char *name) : A függvény bemenete értéke egy string – a felhasználó neve. A függvény először kiírja a bejelentkezett felhasználó ismerőseit a lista függvény alkotta láncolt lista segítségével , akiknek üzenetet tud írni, majd beolvassa kinek szeretne üzenetet küldeni, ezután a messkiir függvény segítségével megjeleníti az eddigi beszélgetéseket, és felajánlja új üzenet írását. Az üzenet elküldése „-” enter el történik. Miután a felhasználó beírta az üzenetet, a messenger függvény átadja az adatokat a sendmess függvénynek, amely elvégzi a file ba írást majd a messkiir függvény segítségével megjeleníti az üzeneteket, immár az elküldött üzenettel együtt. Az egyik ember üzeneteit a jobb olvashatóság miatt „+” al kezdi, a másikat „-” al. Addig ismétlődik ez a while ciklus amíg a felhasználó nem szeretne kilépni, ezután visszatér a függvényből.

void messkiir(char *name1,char *name2): Bemeneti értéke kettő string, az egyik a bejelentkezett felhasználó, a másik az illető akivel beszélni szeretne a felhasználó. A függvény megnyitja az üzenet.txt filet, majd láncolt listát csinál a sorokból, azokat az üzeneteket kiírja a képernyőre ahol a címzett=ismeros és feladó=felhasználó, vagy címzett=felhasználó és feladó = ismerős. Különböző jelekkel kezdve („+” vagy „-”) a jobb olvashatóság érdekében. A végén felszabadítja a foglalt memóriát és bezárja az uzenet.txt filet és visszatér a függvényből.