Programozói dokumentáció

Alapvető cél, működés

A „garage” program egy autószerviz nyilvántartása. A program kezeli az autószerviz ügyfeleit, az ügyfelekhez tartozó autókat és az autókon elvégzett/elvégzendő javításokat. A program a konzolon keresztül kommunikál, fájlból olvas be adatot amennyiben van mentett adat, fájlba írja az új vagy módosított adatot. A 9 különböző menüpont külön modulban van, saját *.c és *.h fájllal. A fő menüpontok moduljain kívül egy main.c, menu.c, menu.h fájl van, a program a main-ből indul és ott ér véget. A main a fájlokból (ugyfelek.txt, autok.txt, javitasok.txt) való beolvasás után meghívja a menüt, ez pedig a kiválasztott menüpont függvényét. Amennyiben bárhol memóriafoglalási vagy fájlelérési probléma van, a program ezt jelzi a felhasználónak, felszabadítja a lefoglalt memóriát és kilép.

Adatszerkezetek

Dátum: év, nap, hónap, nap egészek

Javítás név: tíz lehetséges opció, minden opcióhoz egy egész szám tartozik 1-től 10-ig

Ügyfél: azonosító, vezetéknév, keresztnév, telefonszám, autó tömb (az adott ügyfélhez tartozó autókra mutató pointerek tömbje). Láncolt listába beépíthető (van előző és következő pointer)

Autó: rendszám, aktív státusz, tulajdonos (ügyfélre mutató pointer), márka, javítás tömb (az adott autóhoz tartozó javításokra mutató pointerek tömbje). Láncolt listába beépíthető (van előző és következő pointer)

Javítás: azonosító, autó (autóra mutató pointer), kész státusz, ár, befejezés dátuma (ha még nincs kész, akkor 0/0/0). Láncolt listába beépíthető (van előző és következő pointer)

Ügyfél lista: az első és utolsó ügyfélre mutató pointereket tárolja

Autó lista: az első és utolsó autóra mutató pointereket tárolja

Javítás lista: az első és utolsó javításra mutató pointereket tárolja

Adatok formátuma

Adatbevitel, kijelzett adat: ékezetes karaktert nem használ (csak angol ABC betűi)
Dátum: ÉÉÉÉ/HH/NN
Név: Vezetek Kereszt (1 db vezeték és 1 db keresztnév lehetséges) (angol ABC betűi és ’-’)
Telefonszám: pl: 301234567 (kilenc számjegy)
Ár: pl: 10000, ezt a program a javítás neve alapján megadja
Javítás neve: tíz lehetőség, a felhasználó a kívántnévhez tartozó szám beírásával választ
Javítás állapota: 0/1, alapból 0, amennyiben befejezettnek jelölték 1
Rendszám: nagy betűk és számok
Autó aktivitása: 0/1, alapból 1, amennyiben törölték (azaz deaktiválták) 0

Azonosító: egész szám 1-től kezdődően

Memóriakezelés

A program indulásakor a fájlbeolvasások mindegyikénél új láncolt listát hoz létre (memóriát foglal a lista két végpontára mutató pointernek), amennyiben van beolvasott adat, ezt ügyfelenként/autónként/javításonként beolvassa, memóriát foglal neki és hozzáfűzi a láncolt lista végére, így a lista mérete dinamikusan nő. Új ügyfél/autó/javítás esetén hasonlóan memóriát foglal neki és hozzáfűzi a láncolt lista végére, amennyiben minden adat megfelelő. Bármilyen a program részéről érkező hiba esetén a felszabadít minden foglalt memóriát és kilép. Amennyiben a hiba a felhasználó részéről érkezik, legtöbb esetben dönthet, hogy visszalép a menübe (az aktuális, „feleslegessé vált" memóriát felszabadítva) vagy újra próbálja (a „feleslegessé vált" memóriát hasonlóan felszabadítva).

A láncolt listák felszabadítását három függvény végzi (freeCustomerList, freeCarList, freeRepairList), ezek a main-ben hívódnak meg a program lefutása végén (kijelentkezés vagy memória-, fájlhiba esetén).

Fájlkezelés

A program fájlolvasással kezdődik (readCustomerListFromFile, readCarListFromFile, readRepairListFromFile), amennyiben az ügyfelek fájlt nem tudja megnyitni a program azt és a másik kettőt is üresnek tekinti, a listákat nem tölti fel elemekkel. Hasonlóan, ha nem tudja megnyitni az autók listát, a javítások listát is üresnek veszi. Fájlba írás akkor van, ha új ügyfél/autó/javítás hozzáadása vagy módosítása sikeresen megtörtént, ilyenkor az egész fájlt újraírja (új autó esetén az ügyfelek fájlt is átírja, mivel abban eltárolódik az adott ügyfélhez az új autó, ugyanígy tesz új javítás esetén az autók fájllal). Amennyiben a fájl megnyitásánál hiba lép fel, ezt a program jelzi a felhasználónak és kilépteti magát.

Hibakezelés

Felhasználói hiba esetén a szituációtól függően lehetőség van újra próbálni és visszamenni a főmenübe is, vagy csak újra próbálni lehetséges.

Main

int main(void): meghívja a fájl beolvasó függvényeket, ha van adat, annak memóriát foglal, betölti a fájlból. Ciklusban hívja meg a menüt, az addig fut, amíg kilépés parancsot nem kap. Meghívja a listát felszabadító függvényeket.

Menu

bool menu(CustomerList *customerList, CarList *carList, RepairList *repairList): A három láncolt lista kapja paraméterül, kiíratja a menüt és meghívja a funkcióválasztót. Visszatérési értéke az utóbbi értékétől függ.
int printMenu(): Kiíratja a menüt, bekéri a választott menüpont sorszámát. Visszatérési érteke az utóbbi egész szám.
bool functionSelector(const int menuNumber, CustomerList *customerList, CarList *carList, RepairList *repairList): A három láncolt listát és a választott funkció számát kapja paraméterül,

visszatérési értéke az adott funkciótól függően igaz vagy hamis. A kapott szám alapján hív meg egy funkciót.
int scanDecimal(const int min, const int max): Beolvas egy egész számot a paraméterül kapott minimum es maximum között. Ezt a számot visszaadja, addig fut, amíg a beolvasás nem sikeres.

1. Új ügyfél felvétele

bool newCustomer(CustomerList *customerList): a main hívja meg, paraméterül kapja az ügyfelek listát, visszaadja, hogy a program futhat-e tovább, vagy ki kell léptetni. Új ügyfelet hoz létre.

2. Új autó felvétele

bool newCar(CustomerList *customerList, CarList *carList): a main hívja meg, paraméterül kapja az ugyfelek es autok listat, visszaadja, hogy a program futhat-e tovább, vagy ki kell léptetni. Új autót hoz létre egy adott ügyfélhez.

3. Ügyfél szerinti keresés

bool searchByCustomer(CustomerList *customerList, CarList *carList, RepairList *repairList): a main hívja meg, paraméterül kapja az ügyfelek, autók és javítások listát, visszaadja, hogy a program futhat-e tovább, vagy ki kell léptetni. Egy adott ügyfelet keres meg, ha megtalálta, kiírja a hozza tartozó autókat, azok javításait.

4. Rendszám szerinti keresés

bool searchByPlate(CarList *carList, RepairList *repairList): a main hívja meg, paraméterül kapja az autók és javítások listát, visszaadja, hogy a program futhat-e tovább, vagy ki kell léptetni. Egy adott autót keres meg, ha megtalálta, kiírja a javításokat.

5. Javítás felvétele vagy módosítása

bool repairs (CarList *carList, RepairList *repairList): a main hívja meg, paraméterül kapja az autók és javítások listát, visszaadja, hogy a program futhat-e tovább, vagy ki kell léptetni. Új javítást hoz létre, vagy meglevőt fejez be egy adott autó esetén.

6. Autó törlése

bool deleteCar(CarList *carList): a main hívja meg, paraméterül kapja az autók és javítások listát, visszaadja, hogy a program futhat-e tovább, vagy ki kell léptetni. Meglevő aktív autót deaktivál.

7. Hamarosan lejáró vizsgájú autók listája

bool soonExpires(CarList *carList): a main hívja meg, paraméterül kapja az autók es javítások listát, visszaadja, hogy a program futhat-e tovább, vagy ki kell léptetni. Kiírja a 60 napon belüli lejáratú autókat.

8. Szerviztörténet megjelenítése

bool repairHistory(RepairList *repairList): a main hívja meg, paraméterül kapja az autók és javítások listát, visszaadja, hogy a program futhat-e tovább, vagy ki kell léptetni. Kiírja a kész javításokat időrendi (csökkenő) sorrendben.

0. Kilépés

bool logOut(): a main hívja meg, paraméterül kapja az autók es javítások listát, visszaadja, hogy a program futhat-e tovább, vagy ki kell léptetni (mindig kilépteti). Kilep a programból.