Spertli Krisztián - IYX0EQ

Programozói dokumentáció

Autószervíz

A projekt felépítése

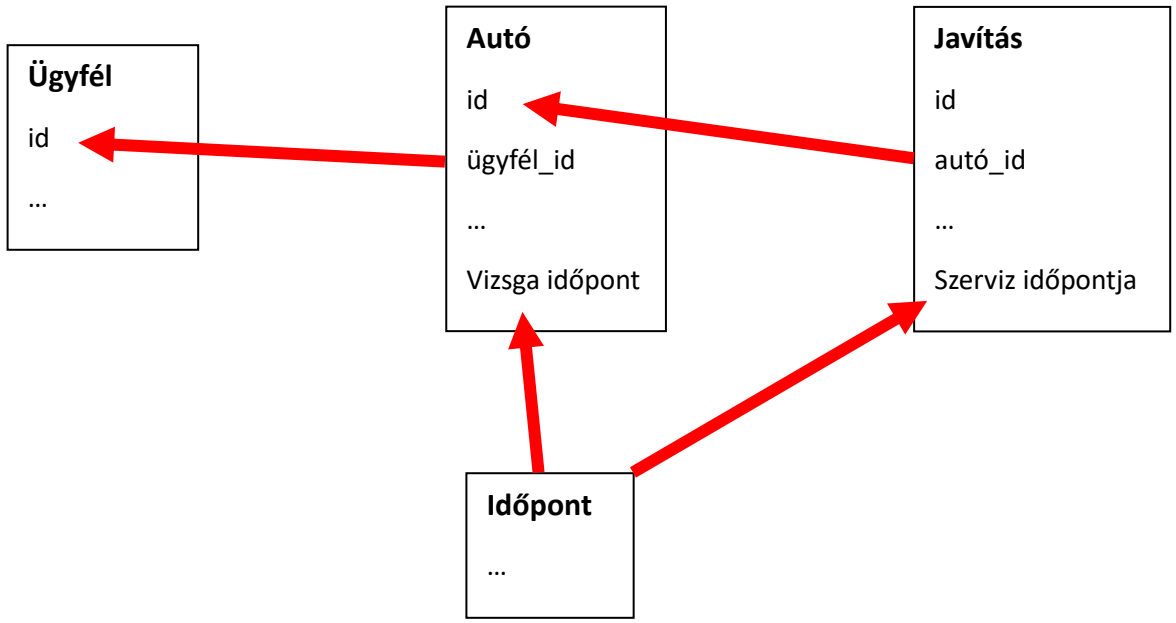
A projekt alapvető felépítése, hogy több fájlból áll, mely a programban a különböző feladatainak elvégzését szolgálja. A main függvények kívül megtalálható három jegyzettömb fájl, mely az adatok tárolását teszi lehetővé. Továbbá megtalálható egy ***adatszerkezet***, ***fajlkezeles***, ***feladatok***, ***menu***, illetve egy ***segedfuggvenyek*** mappa, melyekben .c és .h fájlok helyezkednek el. A programnak külső könyvtárra, illetve grafikus környezetre nincsen szüksége, minden a parancssorba jelenik meg. A projektben megjelenő karakterek ékezetek nélküliek, ennek következtében a beolvasni kívánt karaktersorozatokat is ékezet nélkül várja a program.

Adatszerkezetek

A program adatszerkezete három láncolt listából és egy struktúrából áll. A listák szükségessége az adatok elkülönülését szolgálja, külön lebontva az ügyfelekre, autóikra, és a rájuk vonatkozó javításokra. Láncolt listákra, azért volt szükség, mivel az adatok memóriakezelése könnyebben megoldható, mintha dinamikus tömbök folyamatos átméretezésén kéne gondolkodni, emellett a listaelemek szintén egymás után helyezkednek el, ami az előnyünkre vállhat. A külön elhelyezkedő időpont struktúra elengedhetetlen az autóknál a vizsgaidőpont eltárolására, a javításoknál pedig annak időpontjának leírása miatt.

A különböző listák közötti kapcsolatot azonosítók biztosítják. Az autók adatainak tárolása során szüksége van egy ügyfél id-re, a javításoknál pedig egy autó id-re, hogy pontosan melyik autóhoz is tartozik.

Az eddig leírt kapcsolatokat az alábbi ábrával szemléltetném:



A listában szereplő adatok a feladatnak megfelelően:

Ügyfél: int id, char *név, char *elérhetőség
Autó: int id, int ügyfél_id, char *rendszám, char *típus, Időpont vizsga
Javítás: int id, int autó_id, char *leírás, char *ár, Időpont dátum

A struktúra adatai:

Időpont: int év, int hónap, int nap

Függvények dokumentációja:

Fájlkezelés:

Ugyfel *fajbol_beolvasas();

Feladata:
Megnyitja az ügyfelekhez tartozó szövegfájlt és a benne található adatokat soronként olvassa, míg a fájl végéhez nem ér. Az olvasás során a sorokat dinamikus memóriával foglalja, majd a benne található elemeket a megfelelő láncolt listába helyezi. Az elemek elválasztásért a pontosvessző felel, melyet a függvény figyelembe vesz. A beolvasás végeztével a fájlt bezárja.
Paramétere:
Típusa void.
Visszatérési értéke:
A függvény az adott lista elejére mutató pointert adja vissza, hogy a memóriában tudjuk, hogy a lista hol helyezkedik el.

Auto *fajbol_beolvasas_2();

Az alábbi függvény ugyanúgy működik, mint a fent említett, annyi különbséggel, hogy az ***autókhoz*** tartozó szövegfájlból olvassa be a szöveget.

Javitas *fajbol_beolvasas_3();

Az alábbi függvény ugyanúgy működik, mint a fent említett, annyi különbséggel, hogy a ***javításokhoz*** tartozó szövegfájlból olvassa be a szöveget.

void ugyfelek_mentes(Ugyfel *lista);

Feladata:
Az ügyfelekhez tartozó fájl megnyitás írásra, hogy a láncolt listában lévő adatokat kimásolja a szövegfájlba. Végül a fájlt bezárja.
Paramétere:
Az ügyfél listára mutató pointert kapja meg, hogy tudja honnan kell kimásolni az adatokat.
Visszatérési értéke:
Void típusú, ezért nem ad vissza semmit, csak az adott feladatot elvégzi.

void autok_mentes(Auto *lista);

Az alábbi függvény ugyanúgy működik, mint a fent említett, annyi különbséggel, hogy az ***autókhoz*** tartozó szövegfájlba rögzíti az adatokat.

void javitasok_mentes(Javitas *lista);

Az alábbi függvény ugyanúgy működik, mint a fent említett, annyi különbséggel, hogy a ***javításokhoz*** tartozó szövegfájlba rögzíti az adatokat.

void free_ugyfelek(Ugyfel *ugyfelek);

Feladata:

Amikor dinamikus allokált memóriaterületeket foglaltunk le a listaelemek tárolásához, akkor azokat fel is kell szabadítani, amit az alábbi függvény végez. Végigmegy az ügyfelek listán és felszabadítja a területeket.

Paramétere:

Az ügyfél listára mutató pointert kapja meg, hogy tudja hol kell felszabadítani az adatokat.

Visszatérési értéke:

Void típusú, ezért nem ad vissza semmit, csak az adott feladatot elvégzi.

void free_autok(Auto *autok);

Az alábbi függvény ugyanúgy működik, mint a fent említett, annyi különbséggel, hogy az ***autókhoz*** tartozó listaelemeket szabadítja fel.

void free_javitasok(Javitas *javitasok);

Az alábbi függvény ugyanúgy működik, mint a fent említett, annyi különbséggel, hogy a ***javításokhoz*** tartozó listaelemeket szabadítja fel.

Feladatok:

Létrehozás:

Ugyfel *uj_ugyfel(Ugyfel *lista);

Feladata:

Képes új ügyfélt felvenni a láncolt listában és az ehhez szükséges adatok bekérni. A függvény megvizsgálja a végén, hogy van-e listában elem, ha nincs akkor kezdőelem lesz, ha van, akkor pedig az utolsó elemhez fűzi.

Paramétere:

Az ügyfél listára mutató pointert kapja meg, hogy tudja hova kell behelyezni az adatokat.

Visszatérési értéke:

A függvény az adott lista elejére mutató pointert adja vissza, hogy a memóriában tudjuk, hogy a lista hol helyezkedik el.

Auto *uj_auto(Auto *lista, Ugyfel *eleje);

Az alábbi függvény hasonlóan működik, mint a fenti, annyi különbséggel, hogy most új ***autó*** felvételére kapunk lehetőséget, amit egy adott ügyfélhez tudunk hozzávenni. Mivel ügyfélhez tartozik a jármű, ezért a függvény paraméterébe mindkét listatát be kell vennünk. Visszatérési értéke az autók listára mutató pointer.

Javitas *uj_javitas(Javitas *lista, Auto *eleje);

Az alábbi függvény hasonlóan működik, mint a fenti, annyi különbséggel, hogy most új ***javítás*** felvételére kapunk lehetőséget, amit egy adott autóhoz tudunk hozzávenni. Mivel a javítás egy autóhoz tartozik, ezért a függvény paraméterébe mindkét listatát be kell vennünk. Visszatérési értéke a javítások listára mutató pointer.

Törlés:

Auto *auto_torlese(Auto *lista, int const auto_id);

Feladata:

A listában adott autó azonosító alapján lehet törölni a járművet. Ha megfelelő azonosítót adunk meg, akkor az elemet törli és a maradékot rendezi, hogy ne legyen a listában üres hely.

Paraméterei:

Maga az autókhoz tartozó láncolt lista, illetve egy egész szám, hogy melyik azonosítójú járművet szeretné törölni a felhasználó.

Visszatérési értéke:

A függvény az adott lista elejére mutató pointert adja vissza.

Javitas *javitas_torlese(Javitas *lista, int const auto_id);

A függvény hasonlóan működik a fentihez képest, annyi különbséggel, hogy itt a kívánt autó törlése után, ha a járműhöz tartoztak javítások akkor azokat is törli az id alapján, majd rendeződik a javítás lista. Visszatérési értéke az adott lista elejére mutató pointer.

Keresés:

void ugyfel_autoi_kereses(Ugyfel *lista, Auto *eleje);

Feladata:

Megadott név részlet alapján keres az adatbázisban a függvény, ahol ha egyezést talál valamelyikkel, akkor bizonyos információkat kilistáz táblázatos formában és kiírja a találatok számát is.

Paraméterei:

A fenti két lista a paramétere, mivel ügyfél név alapján keres, majd az ügyfélhez tartozó gépjármű adatok egy részét listázza ki.

Visszatérési értéke:

Void típusú, ezért nem ad vissza semmit, csak az adott feladatot elvégzi.

void rendszam_kereses(Auto *lista);

Hasonlóan működik, mint a név alapú keresés csak itt rendszám alapján történik. Itt csak az autók listája a paraméter, mivel az magába foglalja a járműhöz tartozó ügyfél azonosítóját. Szintán táblázatos formában jelennek meg a megtalált adatok.

void szerviz_tortenet(Javitas *lista);

Hasonlóan működik, mint a fentiek, annyi változtatással, hogy itt az adott autó azonosító által kilistázza az addig folytatott javításokat. Nincs szükség több lista megadására, csak a javításokéra, mivel az magába foglalja a megfelelő adatokat.

void lejaro_vizsga(Ugyfel *lista, Auto *eleje);

A lejáró vizsgák kilistázása is hasonló, itt az ügyfél és autó lista is megvan adva paraméterként, mivel, ha egy járműnek a műszaki vizsgája két évnél régebbi akkor kilistázza az ügyfél nevét és autójának bizonyos adatait szintén táblázatos formában.

Segédfüggvények:

char *hosszu_sort_olvas();

Egy adott sort olvas be a konzolról dinamikus foglalással az új sor jelig, majd visszaadja annak helyét.

char *hosszu_sort_olvas_fajbol(FILE *file);

Hasonlóan működik, mint a fenti függvény csak fájlból olvas be nem pedig konzolról.

Ugyfel *ugyfel_keresese(Ugyfel *lista, int const ugyfel_id);

Az ügyfelek listában megkeresi hogy szerepel-e a megadott id, ha igen akkor visszaadja helyét, különben NULL pointert.

Auto *auto_keresese(Auto *lista, int const auto_id);

Hasonló a fentihez csak most autó azonosítót keresünk.

char *masolas(char const *szoveg);

A függvény az adott szöveget memóriába foglalja és másolja egy átmeneti helyre, majd az átmeneti hely helyét adja vissza.

bool tartalmaz(const char *szoveg, const char *reszlet);

Megvizsgálja, hogy adott szövegben szerepel-e a megadott részlet. Visszatérési értéke igaz vagy hamis. Mindkét szöveget előbb nagybetűsít, hogy a kis- és nagybetűk megkülönböztetésével ne legyen probléma.

Idopont szoveg_idopont_konvertalas(char const *szoveg);

A beolvasás során megadott időpont szöveget átkonvertálja időpont struktúrává, hogy a további feladatok elvégzésekor használni tudjuk.

int nap(Idopont t);

Adott dátumról megmondja, hogy a t.nap része hányadik napja az évnek, figyelembe véve a szökőéveket. Visszatérési értéke ez a napszám érték lesz.

int nap_kulonbseg(Idopont t);

Az előző függvényt felhasználva megállapítja, hogy egy adott dátum a rendszeridőtől számítva, hány napja telt el. Visszatérési értéke ez a különbség lesz.

Menü:

A menü részben csak void típusú függvények találhatóak, melyek csak adott szöveg kiírásért felelnek, köztük a szerviz logója is itt található.

MAIN:

A főprogramban az adott függvények használata valósul meg, ahol egy menürendszer van kialakítva. Kezdetben a main beolvassa a három szövegfájlból az adatokat, ha vannak, majd a bizonyos menüpontokon keresztül hajthatunk végre feladatokat. Kilépéskor a láncolt listában lévő új változtatásokat kimenti a fájlokba, végül a listaelemeket fölszabadítja.

Megjegyzés:

Az adott függvényeken belül, ahol valami megnyitása vagy használata nem sikerül ott külön visszatér más értékkel. Pl. ha nem sikerül fájlt megnyitni akkor NULL a visszatérési érték vagy pl. ha ügyfélnév keresésnél nincs az ügyfélhez feltöltve autó akkor sima return figyelhető meg a kódban.