Programozói dokumentáció

Fazekas Bence Mihály
W0TH54

A programom legalapvetőbb dolga, hogy adatokat legyen képes tárolni, ezeket vissza tudja olvasni parancsra, illetve tudjon keresni és törölni közülük.

A program indításakor egy elágazással választ a felhasználó funkciót, amelynek végrehajtása után egy végtelen ciklus biztosítja, hogy csak egyféleképpen álljon le a programom.

Az adatok felvételét kérdésekre lebontva hajtom végre. A program mindig egyszerre egy pontosított kérdést tesz fel a felhasználónak, majd az adott választ ellenőrzi. (amennyiben a továbbiakban fontos szerepe lesz a kapott információnak, illetve a formátuma szabvány-szerű (pl. Rendszámok felépítése)).

Adatok ellenőrzése:

Ahol szerepelhet 'szóköz' karakter az adatban, ott fájlba íráskor a bemenetek ellenőrzését karakterenként végzem. A feltétel egyszerűen annyi, hogy addig mentsen le minden karaktert a program, amíg 'entert' nem olvas be. Ennek figyelésére az ASCII kódtáblát használom. Az enter kódja jelzi a programnak, hogy véget ért az adattag eleminek a beolvasása.

Vannak olyan bementek, ahol egy vagy több szabály szerint is ellenőrizni szükséges. Itt hosszt, felépítő elemeket, illetve esetleges határokat vizsgálok.

- Az egyik legfontosabb ilyen a rendszámok esetén kerül elő. Ugyanis, egy rendszám (attól függően, hogy mikori) állhat 6, vagy 7 karakterből. Ha több vagy kevesebb a bemenet hossza, akkor az biztosan hibás. A második körben, megvizsgálom a felépítő karaktereket. Ez egy szabvány, ettől nem lehet eltérni, tehát az ellenőrzés alapja szilárd. Ha bármelyik részben elbukik a kapott adott érték akkor azt program elveti és újat kér a felhasználótól, majd azt is aláveti az összes feltételnek.
- Határvizsgálatot hajtok végre az évszámok bekérésénél, hiszen ez pontosan megmondható, hogy mikor készült az első autó, vagy, hogy maximum mennyi időre adnak ki műszaki vizsgát, valamint egyértelmű, hogy hány napos lehet egy adott hónap. (Ha a szökőévet nem vesszük figyelembe) illetve, hogy mennyi hónap van egy évben.
- Tipikusan elemvizsgálatot hajtok végre, amikor a felhasználótól egy egyjegyű számot várok. Nem fogad el a programom sem betűt, sem kétjegyű számot, sem semmilyen más írásjelet. Ezek kiszűrését a következőképpen hajtom végre. Szövegként tárolom az adatokat, majd az ASCII kódtáblát és szövegkezelő függvényeket használok a feltételek megírásához.

Adatok mentése

A helyes inputot azonnal fájlba menti a programom, erről a felhasználót nem kérdezi meg. A fájlokban az összetartozó adatok egy sorban szerepelnek, közéjük a program automatikus elhelyez egy elválasztó karaktert (':'). Ha minden kérdésre válasz érkezett, akkor elhelyezek sor vége jelölést (';') illetve azonnal nyitok egy új sort, amivel a lehetséges következő beírást készítem elő. Az adattagok közötti és a sorvégi karakterek kulcsfontosságúak, hiszen ezek alapján lesz képes a program a tárolt információkat a későbbiekben feldolgozni.

Adatszerkezetek

A fájlban használt szerkezeten kívül struktúrákat, illetve dinamikus tömböket használok. A szervizkönyv létrehozásánál ugyanis elengedhetetlen, hogy minden adatot elérjek az adott járműhöz. Egy autó többször is járhatott a szervizben, ezért dinamikusan foglalt struktúrát használok, hogy minden egyes elemet egyesével meg tudjak vizsgálni.

```
typedef struct Javitas{

  char rendszam[10];
  char ar[100];
  char datum;
  char* leiras;

}Javitas;
```

```
typedef struct Datum
{
   int ev;
   int honap;
   int nap;
}Datum;
```

A másik struktúrám a dátumok kezelésére szétválogatására, ellenőrzésére szolgál.

Dinamikus tömböt használ a programom az olyan adatok tárolására, amelynek nem tudom a hosszát előre. Ilyen például többek között a javítások leírása. Ezeknél az eseteknél, karakterenként olvasok be a felhasználótól és minden egyes beolvasásnál nyújtom a tömbömet. Enter észlelése után (ASCII kód felismerésével) az utolsó üres helyre lezáró nullát helyezek el, hogy a szövegkezelő függvények képesek legyenek hiba nélkül értelmezni az adatokat.

Függvények, eljárások:

Adatfeljegyző függvények:

void Uj_ugyfel(void);

-Az előzőekben leírtak alapján létrehozza és feltölti az Ügyfelek.txt -t

void Uj_Auto(void);

- Az előzőekben leírtak alapján létrehozza és feltölti az Autok.txt -t

void Javitasok_felvetele(void);

-Az előzőekben leírtak alapján létrehozza és feltölti a Javitasok.txt -t

Ellenőrző függvények:

char* ellenorzott evszam bekeres auto felfevetelhez(void)

-A nevéből is adódóan a fentebb leírt módszerekkel egy olyan pointert ad vissza a függvény, ami egy dinamikusan foglalt memóriaterületre mutat, ahol biztosan helyes évszám szerepel az autó műszaki vizsgájának szempontjából. Formailag helyesnek tekintek minden olyan időpontot, ami az adott évbe (2022) esik vagy pedig nem haladja meg több mint 4 évvel ezt. (Ennél több időre nem adnak hivatalosan műszaki vizsgát)

char* ellenorzott_evszam_bekeres_javitas_felfevetelhez(void);

-Nagyban hasonlít a előzőhez, avval a különbséggel, hogy itt minden olyan évszámot tekintek helyesnek ami 2000-nél nagyobb (20 évre visszamenően úgysem szoktak javításokat felvenni) és 2023-nál kisebb.

char belepes_megerosites(void);

-Ez a függvény a félrenyomást hivatott kezelni, vagyis, ha felhasználó olyan funkcióba lépett amit nem szeretett volna. Minden menüpont kiválasztása után egy eldöntendő kérdést jelenik meg amivel vissza lehet lépni a főmenübe, illetve megerősíteni a választást.

char * Rendszambekeres(void);

-A fent leírt rendszámellőrzést hajtja végre. Eredményként egy dinamikusan foglalt memóriaterületre mutató pointer-t ad vissza.

int Hany_sor_van_a_File_ban(FILE* file);

-A fájlkezeléshez, illetve a dinamikus memóriafoglaláshoz több helyen elengedhetetlen, hogy tudjam hány adat (hány sor) van az adott fájlban. Ez a függvényt ezt végzi el, még pedig úgy, hogy megszámolja az „új sor" karaktereket a paraméterében kapott szöveges fájlban. Az eredmény egy int típusú változó.

Kereső, megtekintő, törlő függvények:

void Ugyfelmegtek(void);

-Kiírja a console-ra az Ugyfelek.txt tartalmát

bool Auto_torles(void);

-Kiírja a console-ra az Autok.txt sorait egymás alá sorszámozva, majd a felhasználó által megadott sorszámú kivételével átmásolja őket egy ideiglenes fájlba, evvel lényegében törlést végrehajtva

void Javitas_megtek(void);

-Egy bekért rendszám alapján kiírja a console-re a Javitas.txt-ben szerepelő, adott rendszámhoz tartozó adatokat. Amennyiben nem létezik ilyen, arról is tájékoztatja a felhasználót.

void Kozeli_lejaratu_autok_listazasa(void)

-Kiírja a mindig jelenlegi naptól számított 30 napon belül lejáró, illetve már lejárt műszaki vizsgás autók minden adatát.

void Rendszam_alapu_kereses(void)

-Rendszám alapján keres az Autok.txt-ben, ha egyezést talál kiírja a hozzá tartozó összes adatot

Rajz és menü függvények:

void Logo_Rajz(void);

-Kirajzolja az ASCII art logót

void menukiiras(void);

-Kiírja a menüt, ahonnan lehet funkciót választani

int menupontvalasztas(void);

-Elvégzi az ellenőrzött funkcióválasztást

A függvényeim nagy része nem rendelkezik paraméterrel, hiszen ezek kiirnak, törölnek vagy beolvasnak.

Egyéb, kevésbé fontos, de a megértést segítő kommenteket megtalálhatók a kódban.