Autókereskedés modellezése

A program egy autókereskedés adatbázisát kezeli. A menürendszerben való navigáció után adatbázisban lehet majd keresni, új járművet felvinni, járművet eladni, bevételt számolni a tároláson kívül. Kereséskor több mindent alapján is lehet majd keresni pl.: típus, évjárat férőhely, fogyasztás.

A keresést menürendszeres megoldással tervezem megvalósítani. Ki lehet választani milyen tulajdonság alapján kereshessen a felhasználó, majd a rész szöveget, illetve számot is ki lehet választnai. Ki lehet majd listázni bizonyos tulajdonság(ok) alapján.

Inputként a program két filet fog kapni. Az egyikben a kínálat, a másikban az eladott járművek fognak szerepelni. Outputként a szoftver szintén két filet ad, ezek módosítását.

Tipikusan a standard bemenetről kapja majd a szoftver az autó adatait, ha behozzák eladásra a kereskedésbe. A standard kimenet a menüben való navigálásra, az adatok kiírására és hasonló funkciókra szolgáll.

Autókról eltárolt információk:

- Gyártó
- Típus
- Fogyasztás és ennek egysége (pl.:kilowatt/óra)
- Meghajtás :
    - Benzin
    - Dízel
    - Hybrid
    - Elektromos

Eladás esetén:

- Ár
- Mikor
- Vevő

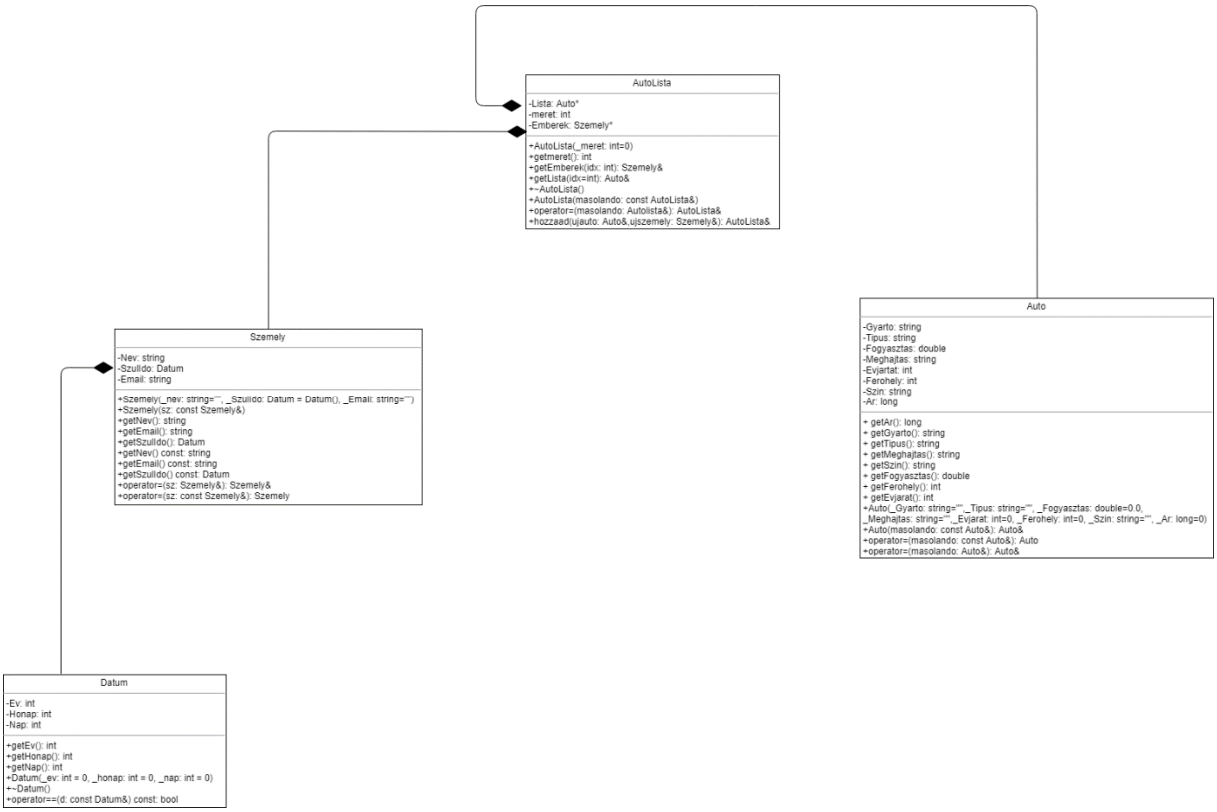
Felvétel esetén:

- Ár
- Mikor
- Eladó

A végleges megoldás az imént leírtatól részben eltérhet.

Somogyi Bence: A programozás alapjai 2 – Nagy házi feladat, pontosított specifikáció.

Osztálydiagram



A megoldásomat draw.io-val készítettem. Jól látható a négy osztály, azok adattagjai, tagfüggvényei és kapcsolatai. A túlterhelt inserter operátorokat az osztályon kívül deklaráltam és definiáltam, ezért nem minősülnek tagfüggvénynek, de természetesen mind a négy osztályra megírtam.

A végleges megoldás ettől részben eltérhet.

Somogyi Bencei: A programozás alapjai 2 – Nagy házi feladat, osztálydiagram.

Felhasználói és Programozói dokumentáció

Felhasználóknak:

A program elsősorban egy autókereskedés alkalmazottjainak szól. Egy adatbázist valósít meg amiben tárolni lehet a kereskedésben szereplő autókat, azok tulajdonosainak adatait illetve a kereskedésből eladott autóknak az adatait és azok előző tulajait.

Mikor a felhasználó elindítja a programot és egy főmenüt talál. Átlátható, hogy hogyan és mivel lehet navigálni a menüpontokban. A program egy számot vár és egy enter leütését.

1- es gombbal a program ki fogja listázni az eladott autók adatait, azok korábbi tulajdonosát.

2- es gombbal az autókereskedésben megtalálható gépjárművek adatait, és jelenlegi tulajdonosait listázza ki a program. A vendégek ezeket az autókat tudják megtekinteni.

3- as gombbal lehet megvalósítani az adatbevitelt. Ekkor az autó adatait és a tulajdonos adatait viszi fel az adatbázisb kínálat listájába.

4- es gombbal a program ki fogja listázni a kínálatot majd egy indexet vár, hogy melyik autót vették meg.

5- ös gombbal a kínálat autói közt lehet keresni minden féle szempont szerint.

6- os gombbal egész egyszerűen bevételt fog számolni a program.

Adatbevitel:

szöveget várnak a következő adattagok: Gyártó,Típus,Szín, Tulajdonos neve és az email címe.

egész számot várnak a következő adattagok: Évjárat, Férőhely és az Ár méghozzá forintban. Számmal kell megadni az ügyfél születési adatait.(Pl.:1987. janár 3. := 1987 1 3 bementetet vár a program.)

Valós számot a fogyasztás adattag vár.

Keresés:

Gyarto szerinti keresésnél egy szöveget vagy szövegrészletet vár a program. Hasonloképpen a Típusra és a Színre való keresésnél.

Fogyasztásnál egy maximális, valós értéket vár a program. A beírt értéknél alacsonyabb fogyasztású értékeket listázza ki a program.

Meghajtásnál érdekes módon egy számot vár a program. Felsorolva látjuk a meghajtást és ezek alapján írjuk be az egész számot. A megfelelően választott erőforrású gépkocsikat írja ki a program.

Az évjáratnál hasonló a helyzet csak fordítva. Senki sem akar régi autót venni. A beírt évjáratnál fitatalabb autókat listázza majd ki a program.

Férőhelynél pontosan tudjuk hány férőhelyes autót akarunk választani. Egy egész számot vár a program.

Ár szerinti keresésnél a maximális árat írja be a felhasználó, az olcsóbb autókat listázza majd ki a program.

Programozóknak:

Az adatszerkezet a fenteb írtak szerint valósítottam meg.

Autó osztály tagfüggvényei:

Az osztály adattagjai ismertek. Minden adattagra van getter függvény. Nem gondolnám, hogy ezt különösebben taglalnni kéne. Visszaadja a nevének megfelelő adattag értékét.

Auto(string _Gyarto="",string _Tipus="",double _Fogyasztas=0.0,string _Meghajtas="",int _Evjarat=0,int _Ferohely=0,string _Szin="",long _Ar=0):Gyarto(_Gyarto),Tipus(_Tipus),Fogyasztas(_Fogyasztas),Meghajtas(_Meghajtas),Evjarat(_Evjarat),Ferohely(_Ferohely),Szin(_Szin),Ar(_Ar){}

Az osztály konstruktora, rendelkezik másoló konstruktorral is. Explicit megvalósított mert nincs dinamikus memóriakezelés.

Auto& operator=(Auto& masolando);

Az osztály értékadó operátora a hármas szabály szerint. Létezik konstans megfelelője is.

std::ostream& operator<<(std::ostream& os,Auto& a);

Nagy szükség van az extraktor operátorra a listázásoknál. Túlterheltem minden osztályra.

Személy osztály tagfüggvényei:

Nem történt változás az adattagokban és

Szemely(string _nev="",Datum _SzulIdo=Datum(),string _email=""):Nev(_nev),SzulIdo(_SzulIdo),Email(_email){}

Az osztály konstruktura. Hasonlóan explicit és létezik másoló konstruktora is.

Megtalálható az adattagok getter függvényei konstans és nem-konstans verziókban.

Szemely& operator=(Szemely& sz);

Szintén a hármas szabály miatt megírtam az osztály értékadó operátorát.

std::ostream& operator<<(std::ostream& os,Szemely& sz);

Itt is megkönnyíti a dolgunkat a túlterhelt extractor operátor.

AutóLista tagfüggvényei:

AutoLista(int _meret=0):meret(_meret){ Lista=new Auto[meret];Emberek=new Szemely[meret];}

A konstruktor itt már dinamikus memóriakezelést is tartalmaz.

Bonyolultabbak az osztály getter függvényei is:

int getmeret(){return meret;}

Igaz ezzel még nincs is gond.

Szemely& getEmberek(int idx){return Emberek[idx];}

Auto& getLista(int idx){return Lista[idx];}

Nos ezek pedig index paramétert várnak. Mint az ismert az adattagok Auto* és Szemely*. Az algoritmusok végigmennek az AutoListán és amikor egy autót és egy tulajdonosát vizsgáljuk akkor egy konkrét autóra és személyre vagyunk kíváncsiak. Ehhez kell az index, hogy a dinamikus tömbböl pontosan egy elemet kapjunk vissza.

~AutoLista();

A destruktor itt azért nem a default mert fel kell szabadítani a dinamikusan foglalt területeket.

AutoLista(const AutoLista& masolando);

A másoló konstruktor.(Különösebb haszna nincs, a hármas szabály miatt.)

AutoLista& operator=(AutoLista& masolando);

Az értékadó operátor.(Különösebb haszna nincs, a hármas szabály miatt.)

AutoLista& hozzaad(Auto& ujauto,Szemely& ujszemely);

A függvény ami hozzáad egy új autót és tulajdonost a listához. Az algoritmus egy „új ” listát ad vissza, amiben egyel több elemet tartalmaz, megnöveli a méretet. Az autó bekerül az adatbázisba. Paraméterül az autót és a tulajdonosát várja. Az adatbekérés és az autóeladás használja.

AutoLista& kivesz(int idx);

A függvény kiveszi az index-edik elemet a listából. Egy számot vár paraméterül. Szintén egy temporális listát generál a függvény. Az algoritmus végigmegy az a listán és beteszi az üres listába az elemeket, akkor ha nem egyezik meg az indexxel a for ciklus számlálója.

std::ostream& operator<<(std::ostream& os,AutoLista& a);

Túlterhelt extraktor operátor visszavezetve az Auto és a Szemely osztályok operátoraira.

fuggvenyek.hpp:

void clearscreen():

Kiír 20 üres sort amivel „törli a képernyőt”.

void Fomenuvalasztkiir():

Sorminta elkerülése végett. Csak kiírja a főmenüben a menüpontokat.

void Filebair(AutoLista& a,const char* filename):

A függvény ami filebaír. Paraméterei a Lista amit fileba kell írni és a file neve.

string elejerolvag(string szoveg, unsigned n) és Datum strtodate(string& str):

A fileból olvasás segédfüggvényei. Az első függvény a paraméterül kapott stringből levág n karaktert majd visszaadja. A másik egy beolvasott stringből csinál dátumot a dátum konstruktorának segítségével.

void Filebololvas(const char* filename,AutoLista& ret):

A függvény a paraméterül kapott fileból olvas és a tartalmat betölti a listába az Auto a személy és az autolista konstruktorainak segítségével. A tipusok konverzióira a strinstreamet használtam.

AutoLista& AdatBeker(AutoLista& a):

Bekéri az auto a szemely adattagjait a standard inputról. Majd a hozzaad tagfügvénnyel hozzaadja azt a listához és visszatér az uj listával.

void Jarmueladasa(AutoLista& Eladva,AutoLista& Kinalat):

Az a függvény csak Átrak egy adatot az egyik listából a másikba. Használjuk a kivesz a hozzaad függvényt. Futás közben kér egy indexet a program és az annyadik elemet törli a kínálatból és rakja az eladottba. Index vizsgálata megvalósított.

void kereses(AutoLista& Kinalat):

A függvény a kínálatban keres külömböző szempontok szerint. Először a szempontot kéri be majd a switch megfelelő ágát hajtja végre. A kilistázás feltételei a felhasználói dokumentációban leírtak szerint. A feltételnek megfelelőket pedig csak kiírja a listából.

long bevetelszamolo(AutoLista& Eladva):

Végig megy az eladott listán, összegezi minden autó árát és visszaadja.