Programozói dokumentáció

A program elemei

A program konzolban lett megvalósítva, így grafikus könyvtárakat nem igényel. A memóriakezelés ellenőrzésére a „debugmalloc.h”-t alkalmazza, melyet az „infoc.eet.bme.hu” oldalról szereztem be.

- main: A két kezdeti menüt tartalmazza amelyekből az összes többi funkció érhető el, valamint a program alapvető működéséhez szükséges „keszlet.txt” és „profilok.txt” fájlok létezését ellenőrzi (és hozza létre őket).
- elad: Az értékesítési módot vezérlő funkciókat tartalmazza.
- keszlet: A készletkezelési módot és a készlet listát vezérlő funkciókat tartalmazza.
- users: A felhasználókezelési módot vezérlő funkciókat tartalmazza.
- util: Fájlkezelő és segédfunkciókat tartalmaz.

A program legfőbb feladatai

Az eladások lebonyolítása és a készletkezelés a program két legkiemeltebben fontos funkciója. Az eladás „rövid parancsokkal” történik, ebbe beleértendő a termékek ID-ja, mely beírásával, és amennyiben darabszám követi azzal, történik. Sztornózáskor ezt a két adatot ’-’ karakter kell kövesse. A ’/’ karakter nyugtaként, a ’\’ karakter számlaként zárja le az eladást. Az eladott termékek csak ekkor kerülnek le a készletről. A készlet nyilvántartása egy külső .txt fájl segítségével és már a programon belül láncolt listával történik. A készletkezelés és a felhasználók kezelése funkciók menükön keresztül lettek megvalósítva.

Valamekkora szintű biztonság fenntartása érdekében a program felhasználókat is nyilvántart a „profilok.txt” fájlban és bejelentkezésüket kéri indításkor. Itt két jogosultsági szintet különböztetünk meg, az eladókét és az „adminisztrátort”. Az eladóknak csakis az értékesítési mód érhető el, míg az adminisztrátor besorolásúaknak teljes hozzáférésük van a program minden funkciójához.

A program által használt adatszerkezetek

A program legfontosabb adata a készlet, melyből duplán láncolt listát épít a hozzátartozó funkciók használatakor, az első és az utolsó elemét is eltárolja. Az utolsó elem nyilvántartása a listához való hozzáadást megkönnyíti. A lista rendezetlen, mivel a program feladatából adódóan nincsem semmiféle rendezettségre szükség.

Használt struktúrák:

- ProductData: Egy termék adatait tartalmazó struktúra
  - char *Id: A termék ID-ja, egy dinamikusan beolvasott string (max 5 char)
  - char *Name: A termék neve, egy dinamikusan beolvasott string (max 20 char)
  - int Count: A termék készleten lévő darabszáma (min 0, max 100.000)
  - int Price: A termék ára (min 0, max 100.000.000)
- StockListElement: A termékekből felépített lista egy elemét tartalmazó struktúra
  - ProductData Product: Egy termék
  - StockListData *Next: A lista következő elemére mutató pointer
  - StockListData *Prev: A lista előző elemére mutató pointer
- UserData: Egy felhasználó adatait tartalmazó struktóra
  - char *username: A felhasználó neve, dinamikusan beolvasott string (max 12 char)
  - char *password: A felhasználó jelszava, dinamikusan beolvasott string (max 24 char)
  - bool admin: értéke true ha adminisztrátor, false ha nem
- BuyerData: Egy számlát igénylő vásárló adatait tartalmazó struktóra
  - char *Name: A vásárló neve, dinamikusan beolvasott string
  - char *Address: A vásárló címe, dinamikusan beolvasott string
  - char* TaxNum: A vásárló adószáma, dinamikusan beolvasott string (max 11 char)

A programban megtalálható függvények

main függvényei

- int main():
  Két fontos bool változót tartalmaz, az isLoggedIn és az isAdmin-t, ezek alapján érhetőek el a program bizonyos funkciói, valamint ez alapján ellenőrzi a program, hogy a felhasználó be van-e lépve.
  Először a StartMenu-t, utána sikeres bejelentkezés esetén a Menu-t hívja meg.
- void StartMenu(bool *isAdmin, bool *isLoggedIn):
  A users-ben található Bejelentkezes-t hívja meg vagy a Kilepes-t, lényegében ez is csak egy menü
- void Menu(bool *isAdmin, bool *isLoggedIn):
  A program legfőbb funkciói innen érhetőek el. A menüpontok kiválasztása függvényében az EladoMod, Keszletkezeles, FelhasznaloKezeles, Kijelentkezes vagy Kilepes függvényt hívja meg.
- void Kijelentkezes(bool *isLoggedIn, bool *isAdmin):
  Megkérdezi a felhasználót biztos-e döntésében, ha igen a két átvett bool-t false-ra állítja.
- void Kilepes():
  Megkérdezi a felhasználót biztos-e döntésében, ha igen kilép a programból

elad függvényei

- void EladoMod():
  Az eladó mód felülete, valamennyi kiírás és a bevitel itt történik.
- void RDisplay(StockListElement *FirstItem):
  Kiírja a képernyőre a felvitt termékek listájának elemeit
- void EvaluateInput(char *input, bool *isDone, bool *openReceipt, StockListElement **FirstItem, StockListElement **LastItem, StockListElement *FirstElement):
  Kiértékeli az EladoMod bemenetét és továbbadja azt valamelyik lentebbi funkciónak.
- void RAdd(StockListElement **FirstItem, StockListElement **LastItem, StockListElement *FirstElement, char *IDin, int cnt):
  Hozzáad egy terméket a felvitt tételek láncolt listájához, ha létezik már benne azt növeli. A cnt ha csak a termék ID-ja lett előzőlegesen beírva 1, másképpen a megadott darabszám.
- void RStorno(StockListElement **FirstItem, StockListElement **LastItem, char *IDin, char *CNin):
  Amennyiben létezik a termék a blokkon, levesz belőle CNin számút.
- void RExit(bool *isDone, bool *openReceipt, StockListElement *FirstElement):
  Visszalép a főmenübe, ha nincsen lezáratlan nyugta/számla.
- void CloseReceipt(StockListElement **FirstItem, StockListElement *FirstElement):
  PrintReceipt-et meghívja, frissíti a készletet és free-eli a blokkot.
- void PrintReceipt(StockListElement *FirstItem):
  Kiírja a képernyőre a blokk elemeit és a végösszeget.

- void CloseInvoice(StockListElement **FirstItem, StockListElement *FirstElement):
  PrintInvoice-ot meghívja, frissíti a készletet és free-eli a blokkot.
- void PrintInvoice(StockListElement *FirstItem):
  Kiírja fájlba a számlát a GetBuyer és GetCurrentDate segítségével
- BuyerData GetBuyer():
  Beolvassa dinamikus stringként a vásárló adatait és visszaadja BuyerData structban azokat.
- void FreeBuyer(BuyerData *Buyer):
  Felszabadítja a Buyer structot.
- void UpdateStock(StockListElement *FirstElement, StockListElement *FirstItem):
  A CloseReceipt vagy CloseInvoice függvények tartalmazzák, frissíti a beolvasott készletet az eladott tételek függvényében.
- char *GetCurrentDate():
  A számla kiírásához használandó, létrehozza a path-jét a mai dátummal percre pontosan a fájl neveként.

keszlet lényeges függvényei

- void KeszletKezeles():
  A készletkezelés menüjét jeleníti meg, valamint bemenetet kér a menüpont kiválasztására
- void StockView(StockListElement *FirstElement):
  Kiírja a képernyőre a készlet elemeit
- void StockAdd(StockListElement **LastElement, StockListElement **FirstElement):
  A termékek hozzáadását kezelő felület/menü
- void SAddID(ProductData *newProduct, StockListElement *FirstElement):
  Beolvas egy termék ID-t és amennyiben szabályos és egyéni, eltárolja a newProduct-ba.
- void SAddName(ProductData *newProduct):
  Beolvas egy termék nevet-t és amennyiben szabályos, eltárolja a newProduct-ba.
- void SAddCount(ProductData *newProduct):
  Beolvas egy termék darabszámot és amennyiben szabályos, eltárolja a newProduct-ba.
- void SAddPrice(ProductData *newProduct):
  Beolvas egy termék árat és amennyiben szabályos, eltárolja a newProduct-ba.
- void SAddSave(StockListElement **FirstElement, StockListElement **LastElement, ProductData *newProduct):
  Elmenti az új terméket a listába és kiírja fájlba azt.
- void SAddExit(ProductData *newProduct):
  Kilép a főmenübe és elveti a beolvasott adatokat
- void StockEdit(StockListElement *FirstElement):
  Beolvas egy termék ID-t és amennyiben létezik megnyitja az SEditMenu-t
- void SEditMenu(StockListElement *ToEdit, StockListElement *FirstElement):
  A termékek szerkesztését kezelő menü

- void SEditID(StockListElement *ToEdit, StockListElement *FirstElement):
  Beolvas egy termék ID-t, amennyiben szabályos és egyéni a ToEdit pointerben megváltoztatja azt
- void SEditName(StockListElement *ToEdit):
  Beolvas egy termék nevet, amennyiben szabályos a ToEdit pointerben megváltoztatja azt
- void SEditCount(StockListElement *ToEdit):
  Beolvas egy termék darabszámot, amennyiben és egyéni a ToEdit pointerben megváltoztatja azt
- void SEditPrice(StockListElement *ToEdit):
  Beolvas egy termék áratt, amennyiben egyéni a ToEdit pointerben megváltoztatja azt
- void StockDelete(StockListElement **FirstElement, StockListElement **LastElement):
  Beolvas egy termék ID-t és amennyiben létezik kitörli azt a SDelAndMend segjtségével
- void SDelAndMend(StockListElement **FirstElement, StockListElement **LastElement, StockListElement *ToDelete):
  A kitörlendő elemet felszabadítja és a lista elemeit szükség esetén összekapcsolja
- StockListElement *ReadStockIntoList(bool *isDone, StockListElement **LastElement):
  A keszlet.txt beolvasása okozta problémákat próbálja kezelni. Az SListBuild-et meghívja és visszaadja az általa épített lista első és utolsó elemének pointerét
- StockListElement *SListBuild(char* filename, StockListElement ***LastElement):
  Felépíti a StockListElement típusokból álló listát és visszaadja az első és utolsó elemére mutató pointert.
- StockListElement *SListAllocate(char *buffer):
  A beolvasott keszlet.txt sort szétszedi, mallocolja, eltárolja egy StockListElementbe és visszaadja azt
- StockListElement *SListFind(StockListElement *FirstElement, char* Id):
  Megkeres ID alapján egy terméket a FirstElement által megadott listában és visszaadja az arra mutató pointert
- void SListFreeList(StockListElement *FirstElement):
  Felszabadítja a FirstElement által megadott StockListElement típusú listát
- int SListWriteToFile(StockListElement *FirstElement):
  Kiírja fájlba a megfelelő formátumban a készlet listát

users lényeges függvényei

- void Bejelentkezes(bool *isAdmin, bool *isLoggedIn):
  A bejelentkezés folyamatát vezeti végig, bekéri a felhasználó nevét, jelszavát. Amennyiben ezek helyesek az isLoggedIn-t true, az isAdmin-t a felhasználó jogosultságától függő értékkel adja vissza.
- UserData *ReadUser(char *input, char *filename):
  Beolvas fájlból egy felhasználót és beteszi egy UserData struktúrába és visszaadja azt.
- void FelhasznaloKezeles():
  A felhasználó kezelés menüpont menüje, innen érhetőek el a lentebbi funkciói
- void UserView(char *filename):
  Kiírja a képernyőre a profilok.txt-ben található felhasználókat

- void UserAdd(char *filename):

  A profilok.txt fájl végéhez hozzád egy új felhasználót
- void UserEdit(char *filename):

  Elkezdi beolvasni és kiírni fájlba a felhasználókat, ha megtalálja a keresettet megnyit egy szerkesztési menüt. Ha végzett a kezelő kiírja a maradékot a fájlba.
- void UserDelete(char *filename):

  Elkezdi beolvasni és kiírni fájlba a felhasználókat, ha megtalálja a keresettet átugorja, ezzel kitörli azt.
- char *UserSearch(char *filename, char *Username):

  Megkeres egy felhasználót és a beolvasott sort visszaadja

util lényeges függvényei

- void CheckDefaultFiles():

  Ellenőrzi a „szamlak” mappa, „profilok.txt” és „keszlet.txt” fileok létezését. Ha bármelyik nem létezik vagy szabálytalanul üres, létrehozza. A profilokat „admin;admin;1” tartalommal.
- bool DoesFileExist(char *filename):

  Megnézi a filename-ként megadott file létezik-e
- bool DoesFolderExist(char *folderPath):

  Megnézi létezik-e a folderPath-ként megadott mappa
- void CreateFolder(char *folderPath):

  A folderPath-ként megadott mappát létrehozza
- bool isFileEmpty(char *filename):

  Megnézi egy fájl első karaktere EOF-e, ha igen true, ha nem false a visszatérési értéke
- int GetFileLength(char *filename):

  A filename-ként meghatározott file sorainak számát adja vissza.
- char *ReadDynStr():

  Beolvas egy stringet dinamikusan és visszaadja a rámutató pointert. Hibák esetén NULL-t.
- char *ReadLimDynStr(int limit):

  A fentebbivel megegyezik funkciója, annyi eltéréssel, hogy a limit-ként megadott karakterszámot olvassa be legfeljebb.