Nagy házi feladat Dokumentáció

Étterem: Dolgozói adminisztrációs program

Labor vezető: Gera Dóra

Felhasználói dokumentáció

A program:

Az alkalmazás a fordító által generált „bin/Debug/*etterem_program_xxcayj.exe*” fájl által indítható. További követelmény, hogy az exe fájl mellett kell elhelyezni azt a „*files*” mappát ahová menteni szeretnénk a program által használt bemeneteket és kimeneteket. Ha a fordítóból szeretné futtatni a programot az imént említett mappának a projektfájl mellett kell lennie. A programnak egyéb kiegészítőkre vagy fájlokra nincs szüksége. Érdemes alkalmazni rendszergazdai módban történő indítást is.

Cél:

A program egy parancssori alkalmazás, melynek célja, egy étterem belső, dolgozói adminisztrációs rendszerének modellezése.

Bemeneti adatok:

Az alkalmazás megfelelő működéséhez szükséges bemeneti adatok a következőek:

Jelzések*: szám típusú*, szöveg típusú (ez egységesen max. 50 karakter/adat)

- **Nyitvatartási idő – „openinghours.txt”**: Célja, a program csak akkor él, ha az étterem nyitva van, ha nincs nyitva, akkor ezt jelzi a felhasználó felé. Amennyiben nem adjuk meg, akkor a program opciói nem elérhetőek. Ez felvihető a megfelelő file beolvasásával, vagy manuálisan is.
  - **Formátum**: *kezdőóra*:*kezdőperc*-*végóra*:*végperc*;\n –ha nyitva van
    ZARVA;\n –ha nincs nyitva
  - Az adatok sorrendje határozza meg a napot. Vasárnaptól-szombatig soronként egy nap.
- **Asztalok – „tables.txt”**: Az étteremben található asztalokat írja le. A program elindul nélküle, de figyelmeztet, hogy ha nem adjuk meg, akkor nem fog megfelelően működni. A megadás történhet a file létrehozásával, majd a program újraindításával, vagy a 1. menüpont 2 –es almenüpontjának kiválasztásával, kézzel.
  - **Formátum**: *asztalkódja*:*hány_fő_fér_el*:*szabad*?;\n
  - Az adatok megadásánál a felhasználó feladata figyelni, hogy van-e már ilyen kódú asztal. Ha mégis bekerül több egyforma adat, akkor a műveletek az első ilyen elemre teljesülnek majd.
- **Menü – „menu.txt”:** A rendelés felvételéhez szintén elengedhetetlen adat a menü. Az asztalokhoz hasonlóan olvasható be. (2. menüpont 2-es almenüpont).
  - **Formátum**: *menükódja*:típus:név:*ár*;\n
  - Az adatok megadásánál a felhasználó feladata figyelni, hogy van-e már ilyen kódú menü. Ha mégis bekerül több egyforma adat, akkor a műveletek az első ilyen elemre teljesülnek majd.
- **Foglalások – „reservation.txt”:** A sikeres előrefoglalásokat menti ide a program, ha már létezik a file, akkor beolvassa, és ez alapján lefoglalja az asztalokat. Ha nem adjuk meg indítás előtt a program probléma mentesen fut.
  - **Formátum**: *asztalkód*:felhasználó_név:*nap*:*itt_van_már?*:*kezdőóra*:*kezdőperc*:*végóra*:*végperc*;\n
  - Nap: foglalás napja = 0-6, vasárnap óta eltelt napok száma.
  - Itt van már?: 1=igen, 0=nem
  - Kézzel beolvasás esetén az asztalkód és az itt_van_már? elhagyandó.

- Az adatok megadásánál a felhasználó feladata figyelni, hogy van-e már ilyen kódú menü. Ha mégis bekerül több egyforma adat, akkor a műveletek az első ilyen elemre teljesülnek majd.

Console-ról való adatmegadás esetén a beolvasás enter jelig megy, azaz az adatokat egy sorba kell írni, a megfelelő formátum megtartásával, a kért elválasztó jellel. (;)

A bemenő adatok beolvasásakor a megfelelő formátumról a felhasználónak kell gondoskodnia, amennyiben egy adott adat nem helyes formátumban van, azt a program nem dolgozza fel, ami hibát eredményezhet. Különösen ügyelni kell a fent említett bemenetek file formájában történő megadásakor, ilyenkor a hibalehetőségek száma megnő.

A program továbbá több helyen is kér be a felhasználótól számadatot, (menüpont, asztalszám, kód…stb.) de ezeket minden esetben egyértelmű módon teszi. Ilyenkor egy specifikus számot, majd azt követő entert vár a szoftver.

Egyetlen speciális eset: Rendelés felvétele menüpont alatt a program speciális formátumú bemenő adatot vár. Ez azon ételek kódja, amelyet a vendég rendelni szeretne. A beolvasás itt is enterig megy. Formátuma: `menükód1;menükód2;…`

Elérhető funkciók menüje:

Ha a bemeneti adatok mindegyike kielégítő, a következő funkciók érhetőek el a felhasználó számára:

- **1=Asztalok kezelése**
  - **1=Asztalok listázása** minden elem
  - **2=Asztal hozzáadása** több elem egyszerre
  - **3=Asztal törlése** egy elem egyszerre
  - **0= Kilepés**
- **2=Menü/menüelem kezelése**
  - **1=Menü listázása** minden elem
  - **2=Menü hozzáadása** több elem egyszerre
  - **3=Menü törlése** egy elem egyszerre
  - **0= Kilepés**
- **3=Új asztal nyitása** egy elem egyszerre
- **4=Rendelés felvétele**
  - **1= Rendelés kiíratása** adott asztalhoz minden rendelés
  - **2= Rendelés felvétele** több elem egyszerre, de egy asztalhoz
  - **3= Rendelés törlése** teljes rendelés törlése, adott asztalhoz
  - **0= Kilepés**
- **5=Számlázás** adott asztalhoz
- **6=Új előrefoglalás rögzítése**
  - **1=Új előrefoglalás rögzítése** adott asztalhoz
  - **2=Létező foglalás beváltása** adott asztalhoz
  - **3= Létező foglalás törlése** adott asztalhoz
  - **4=Foglalások listázása** minden elem
  - **0=Kilepés**
- **0= Kilepés**

Kimeneti adatok:

Az alkalmazás megfelelő működése esetén a kimeneti adatok a következőek:
(A formátumok megegyeznek a bemeneti adatokéval, de minden elválasztó jel után sortörés következik.)

Fájlok:

- **Nyitvatartási idő – „openinghours.txt”**: Tartalma nem változtatható meg a program futása közben, csak abban az esetben, ha helytelenül, vagy nem adott meg bemenetként ilyen fájlt. Kézzel és fájlból való megadás esetén is kimentődik az adat.

- **Asztalok – „tables.txt”**: A leggyakrabban módosított kimeneti fájl. Tartalma frissül minden foglalás előrefoglalás és számlanyomtatás esetén. Valamint az 1. menüpont 2-es és 3-as almenüpont esetén is.
- **Menü – „menu.txt”:** Akkor frissül a tartalma, ha a 2. menüpont 2-es és 3-as almenüpontját használjuk.
- **Foglalások – „reservation.txt”:** Akkor változik meg a tartalma, ha sikeresen felvettünk, töröltünk vagy módosítottunk előrefoglalást.

Console-ra kiírt adatok:

- **Asztalok** – 1-1 menüpont választása esetén
- **Menü** – 2-1 menüpont választása esetén
- **Szabad asztalok** – 3 menüpont választása esetén
- **Rendelés egy adott asztalhoz** – 4-1 menüpont választása esetén
- **Számla egy adott asztalhoz** – 5 menüpont választása esetén
- **Előrefoglalások** – 6-4 menüpont választása esetén

Egyéb információk:

- **Új asztal nyitása** – A program nem végez automata asztalkiosztást, hogy a vendég eldönthesse hová szeretne ülni, azonban választani csak a szabad asztalokból tud.
- **Előrefoglalások** – Lehetséges, hogy egy előrefoglalás már lejárt, ezt a program nem tudja ellenőrizni, így a felhasználó (pincér) feladata, hogy a 6-3-as menüpont segítségével törölje a nem kívánt foglalást, az asztal felszabadítást a program elvégzi ezek után.
- **Rendelés** – A rendelések nem mentődnek ki fájlba, így azt áramszünet esetén újra fel kell venni. Rendelést csak meglévő, foglalt asztalhoz lehet leadni.
- **Számla** – A számla nyomtatásával a vendég azt jelzi, hogy távozik, ezért a rendelése és az asztalfoglalása is törlődik.
- **Asztal, Menu, Előrefoglalás hozzáadása:** Ezen menüpontok esetén a program nem vizsgálja, hogy a hozzáadandó elem kódjával már van-e ilyen elem. Ha ez fontos a felhasználó számára, akkor az ő feladata úgy megadni az adatokat, hogy ne legyen egyezés.

Programozói dokumentáció

Környezet:

A program a lenti feltételek mellett készült, és lett tesztelve, ezért ajánlott ezeket használni.

- Programozási nyelvek:
  - C
- Programok:
  - CodeBlocks 17.12
  - GNU GCC Compiler
- Tesztelt operációs rendszer
  - Windows 10 home
- Perifériák: (már futó program estén)
  - Monitor
  - Billentyűzet

Cél:

A program egy parancssori alkalmazás, melynek célja, egy étterem belső, dolgozói adminisztrációs rendszerének modellezése.

Fájlok:

Az alkalmazás fájljait tároló mappa felépítése a fordítóból történő futtatás után a következőképpen néz ki:

`\bin\Debug\etterem_program_xxcayj.exe` –futtatható kód

`\files\menu.txt` – menüt tartalmazó be és kimeneti fájl

`\openinghours.txt` – nyitvatartást tartalmazó be és kimeneti fájl

`\reservation.txt` – előrefoglalásokat tartalmazó be és kimeneti fájl

`\tables.txt` – asztalokat tartalmazó be és kimeneti fájl

`\files\...` – pontosan ugyanaz mint a \bin\Debug\files mappa tartalma

`\obj\Debug\... .o` – modulok objektum kódjai (azok, amikhez tartozik .c fájl)

`\debugmalloc.h` – modul head fájlja

`\debugmalloc-impl.h` – modul head fájlja

`\etterem_program_xxcayj.cbp` – projek fájl

`\ etterem_program_xxcayj.depend` – projekt fájl kiegészítője

`\etterem_program_xxcayj.layout` – projekt fájl kiegészítője

`\iofunctions.c` – modul C kódja

`\iofunctions.h` – modul head fájlja

`\main.c` – modul C kódja

`\menu.c` – modul C kódja

`\menu.h` – modul head fájlja

`\orders.c` – modul C kódja

`\orders.h` – modul head fájlja

`\reserv.c` – modul C kódja

`\reserv.h` – modul head fájlja

`\tables.c` – modul C kódja

`\tables.h` – modul head fájlja

`\timemanage.c` – modul C kódja

`\timemanage.h` – modul head fájlja

Modulok:

A program a következő modulokból épül fel:

- **iofunctions** – Fájlba írást, fájlból olvasást, console-ról formátum sztring bekérését végző modul, amit az összes többi is használ.

- **main** – Főprogram, ez jeleníti meg a főmenüt, itt található a számlázást végző függvény is. Valamint ehhez van hozzákapcsolva a többi modul közvetlenül.
- **menu** – Teljes 2. menüpont kezelését végző modul. Menü kezelésére alkalmas. Használja az iofunctions-t.
- **orders** – Használja a menu és table modult. Teljes 4. menüpont kezelését végző függvény. Rendelések kezelésére szolgál.
- **reserv** – Előrefoglalások kezelésére alkalmas modul. Használja a tables és a timemanage modult.
- **tables** – Teljes 1. menüpont kezelését végző modul. Asztalok kezelésére alkalmas. Használja az iofunctions-t. Függvényeit más modulok főként foglalások készítésekor használják.
- **timemanage** – Nyitva tartás kezelésére alkalmas modul. Itt találhatóak azok a függvények, amik a nyitva tartás figyelését végzik. Azonban adatstruktúrája alkalmas a reserv modul foglalási időpontjainak tárolására is.

Függvények-adatszerkezetek részletesen (modulokra bontva):

Az alkalmazás működésért a következő függvények felelősek:

A program minden fő adatszerkezetnek láncolt listát alkalmaz, mert a sorrend nem fontos, nem tudni előre mennyi rendelés lesz és könnyű benne keresni, elemeket törölni.

iofunctions

- **char *readFile (int * startsize, char const * file_name):** Általános fájlbeolvasó függvény. Visszatér a beolvasott fájl teljes tartalmával egy tömbben. Bemeneti paramétere egy egész szám, ami a fájlban tárolt szöveg méretét adja meg és egy konstans fájlnév. Előbbit cím szerint veszi át, mert a végén a benne tárolt érték megegyezik majd a valódi hosszával a sztringnek. A startsize nem lehet NULL, vagy <=0.
- **char *readConsole (int * startsize):** Általános beolvasó függvény. Visszatér a Console-ról beolvasott szöveg teljes tartalmával egy tömbben. Bemeneti paramétere egy egész szám, ami a beolvasandó szöveg méretét adja meg. Ezt cím szerint veszi át, mert a végén a benne tárolt érték megegyezik majd a valódi hosszával a sztringnek. Egyik paraméter sem lehet NULL, vagy <=0.
- **void makeFile (char * str, char const * file_name):** Általános fájlba író függvény. Bemeneti paramétere egy karakter tömb, amiben a sorok pontosvesszővel vannak elválasztva. Valamint egy fájlnév. Előbbinek be kell tartani a pontosvesszős formátumot, vagy a fájl üres lesz. Nincs visszatérési értéke.

main

- **int main (void):** Főprogram, nincs paramétere. Visszatérési értéke 0, ha nincs hiba a futás közben. Itt történik a bemeneti fájlok beolvasása, annak levizsgálása, hogy az étterem nyitva van-e, a fő adatszerkezetek példányosítása, ami az egész futási idő alatt tárolja majd az adatokat. Valamint az jeleníti meg a főmenüt is.
- **void printCheck (Orders* orders_list, int table_number, Menu* menu_list):** 3. menüpont megvalósító, számlázó függvény. Nincs visszatérési értéke. A bemenő paraméterei egy Orders típusú láncolt lista első eleme, egy egész asztalszám és egy Menu típusú láncolt lista első eleme. Ezek alapján megkeresi azt a rendelést, ahol az asztal szám meggyezik a paraméterben megadottal, ha megtalálta kiírja az elemeit és az árak összegét, majd törli a rendelést. (3. menüpontba való visszatérése után, még az az asztal, ahol ez a rendelés volt felszabadul.)

menu

- **typedef struct Menu..:** Egy olyan egyszeresen láncolt lista**,** amely tartalmaz egy egész változót, ami az adott menü elem kódját tartalmazza, két, maximum 50 karakter hosszú sztringet (+1 a lezáró nulla). Ezek a menü elem nevét és típusát tárolják el. Továbbá az adott elem árát tároló egész változót. Mivel ez egy láncolt lista, létezik még benne egy következő elemre mutató pointer is.
- **Menu * processMenu (char * base_str, Menu * final_menu_list):** Menu típusú láncolt lista építő függvény. Menu típusú elemre mutató pointerrel tér vissza, ami a benne épített lista első

elemére mutat. Paraméterei, a feldolgozásra váró sztring és az a Menu típusú láncolt lista, amiben esetleg már lehetnek elemek. Itt bármelyik paraméter lehet nulla, mert akkor a függvény NULL pointerrel tér vissza.

- **char * makeStrFromMenu (Menu* first_elemnt):** Ez a függvény visszacsinálja, amit a processMenu függvény csinált. Erre a fájlba íráskor van szükség. Tehát a paraméterben kapott listát sztringgé alakítja, majd visszaadja azt. Lehet NULL, ekkor a sztring is üres lesz.
- **Menu* removeMenuElement (Menu* first, int const code):** Ez a függvény eltávolítja a paraméterben megadott konstans egész szám értékével megegyező asztalszámú listaelemet, majd visszatér a lista első elemére mutató pointerrel. A lista Menu típusú, és a code-dal együtt lehet nulla, mert olyankor nem történi semmi.
- **Menu* isExistMenu (Menu * first_element, int const menu_number):** A paraméterben kapott Menu típusú lista és a konstans egész asztal kód alapján visszatér annak az elemnek a helyével, ahol annak a kódja megegyezik a megadott kóddal. Ha nincs ilyen elem, akkor NULL-al tér vissza.
- **void freeUpMenu (Menu* first):** Felszabadítja a paraméterben kapott Menu típusú láncolt listát. Visszatérési értéke nincs.
- **Menu* manageMenuMP (Menu * first_element):** A 2. menüpont almenüjének megjelenítésére és működtetésére szolgáló függvény. Kitudja írni a paraméterben kapott Menu típusú láncolt lista elemeit, bővíteni tudja azt, valamint törölni tud elemeket. Kilépésnél visszatér a módosított lista első elemére mutató pointerrel. Módosítja a menu.txt fájlt, minden listaváltozásnál.

orders

- **typedef struct Orders…:** Egy olyan egyszeresen láncolt lista**,** amely tartalmaz egy dinamikus méretű egész számokból álló tömböt. Ebben tárolódnak el a rendelt menü elemeket azonosító kódok. Továbbá egy egész változót, ami az előbb említett tömb méretét adja meg, és egy másikat, ami a rendeléshez tartozó asztalszámot tárolja. Mivel ez egy láncolt lista, létezik még benne egy következő elemre mutató pointer is. Egy asztalhoz tartozó rendelésből a listában csak egy lehet.
- **Orders * findOrderByTable (Orders* ordr_list, int const table_number):** Ez a függvény megkeresi azt az elemet a paraméterben kapott Orders típusú listában, ahol az asztalszám megegyezik a kapott konstans egész számmal. Majd visszatér vele. Ha nincs ilyen elem NULL –al tér vissza.
- **Orders * addOrder (char * base_str, Orders *current_element):** A paraméterben kapott aktuális listaelemre mutató pointer rendeléseket tartalmazó tömbjét bővíti a szintén paraméterben kapott sztring alapján. Módosul vele a tömb méretét tároló n változó értéke is. Majd visszatér a megváltozott elemre mutató pointerrel.
- **Orders * processOrder (char * base_str, Orders * order_list, int const table_number):** Orders típusú láncolt lista építő függvény. Orders típusú elemre mutató pointerrel tér vissza, ami a benne épített lista első elemére mutat. Paraméterei, a feldolgozásra váró sztring és az a Orders típusú láncolt lista, amiben esetleg már lehetnek elemek. Itt bármelyik paraméter lehet nulla, mert akkor a függvény NULL pointerrel tér vissza. Továbbá van egy konstans asztalszámot tároló egész típusú paramétere is, ami egy létező asztal száma kell, hogy legyen.
- **void printOrderByName (Orders* current_element, Menu* menu_list):** A paraméterben kapott Menu lista alapján kiírja a képernyőre azokat a rendelés elemeket és adataikat, ahol a kapott rendelésre mutató pointer ételeket tároló tömbben tárolt elem kódja egy létező ételre hivatkozik. Ahol nem ott csak azt írja ki, hogy nincs ilyen étel. Nincs visszatérési értéke. A paraméterben kapott értékek nem lehetnek nullák.
- **Orders* removeOrdersElement (Orders* first, int const code):** Ez a függvény eltávolítja a paraméterben megadott konstans egész szám értékével megegyező asztalszámú listaelemet, majd visszatér a lista első elemére mutató pointerrel. A lista Orders típusú, és a code-dal együtt lehet nulla, mert olyankor nem történi semmi.
- **void freeUpOrders (Orders* first):** Felszabadítja a paraméterben kapott Orders típusú láncolt listát. Visszatérési értéke nincs.

- **Orders * manageOrdersMP (Orders * o_first, TableList* t_first, Menu* m_first):** A 4. menüpont almenüjének megjelenítésére és működtetésére szolgáló függvény. Asztalszámtól függően ki tudja írni a paraméterben kapott Orders típusú láncolt lista elemeit, bővíteni tudja azt, valamint törölni tud elemeket. Kilépésnél visszatér a módosított lista első elemére mutató pointerrel. A TableList és Menu típusú listákra a felhasznált függvények miatt van szüksége. Például csak létező, foglalt asztalhoz vehetünk fel rendelést.

reserv

- **typedef struct Reservation..:** Egy olyan egyszeresen láncolt lista**,** amely tartalmaz egy egész változót, ami az adott asztal kódját tartalmazza, egy maximum 50 karakter hosszú sztringet (+1 a lezáró nulla). Ez a foglaló felhasználónevét tárolja el. Továbbá a foglalás időadatait tartalmazó Time struktúra típusú változót. Mivel ez egy láncolt lista, létezik még benne egy következő elemre mutató pointer is.
- **Reservation * processReservation (char * base_str, Reservation * final_reserv_list, int const table_number, TableList * table_list):** Reservation típusú láncolt lista építő függvény. Reservation típusú elemre mutató pointerrel tér vissza, ami a benne épített lista első elemére mutat. Paraméterei, a feldolgozásra váró sztring és az a Reservation típusú láncolt lista, amiben esetleg már lehetnek elemek. Itt bármelyik paraméter lehet nulla, mert akkor a függvény NULL pointerrel tér vissza. További paraméterei egy konstans egész, ami a foglalni kívánt asztal számát tartalmazza, valamit az össze asztalt tartalmazó lista első elemére mutató pointer. Ez azért kell, hogy sikeres adatmegadás esetén ténylegesen le lehessen foglalni az asztalt.
- **char* makeStrFromReservation (Reservation* first_element):** Ez a függvény visszacsinálja, amit a processReservation függvény csinált. Erre a fájlba íráskor van szükség. Tehát a paraméterben kapott listát sztringgé alakítja, majd visszaadja azt. Lehet NULL, ekkor a sztring is üres lesz. Az asztalok foglaltságán viszont nem változtat.
- **Reservation* removeResElement (Reservation* first, int const code):** Ez a függvény eltávolítja a paraméterben megadott konstans egész szám értékével megegyező asztalszámú listaelemet, majd visszatér a lista első elemére mutató pointerrel. A paraméterben kapott lista Reservation típusú, és a code-dal együtt lehet nulla, mert olyankor nem történi semmi.
- **Reservation* setIsAppropriate (Reservation * res_list, int const code, bool const value):** A paraméterben kapott Reservation típusú lista és a konstans egész asztal kód alapján beállítja annak az elemnek az is_appropriate (jelen esetben itt van-e már?) értékét a konstansként kapott bool értékre. Ha nincs ilyen elem, akkor NULL-al tér vissza.
- **void freeUpReserv (Reservation* first):** Felszabadítja a paraméterben kapott Reservation típusú láncolt listát. Visszatérési értéke nincs.
- **Menu* manageMenuMP (Menu * first_element):** A 6. menüpont almenüjének megjelenítésére és működtetésére szolgáló függvény. Kitudja írni a paraméterben kapott Reservation típusú láncolt lista elemeit, bővíteni tudja azt, be tudja állítani, hogy a vendég megérkezett-e már, valamint törölni tud elemeket. Kilépésnél visszatér a módosított lista első elemére mutató pointerrel. Módosítja a reservetion.txt fájlt, minden listaváltozásnál. A TableList típusú lista paraméterre az asztalok tényleges foglalása miatt van szükség.

tables

- **typedef struct TableList..:** Egy olyan egyszeresen láncolt lista**,** amely tartalmaz egy egész változót, ami az adott asztal kódját tartalmazza. Továbbá az adott asztal kapacitását tároló egész változót és egy logikai változót is ami azt jelzi, hogy az adott asztal éppen szabad-e. Mivel ez egy láncolt lista, létezik még benne egy következő elemre mutató pointer is.
- **TableList * processTables (char * base_str, TableList * final_table_list):** TableList típusú láncolt lista építő függvény. TableList típusú elemre mutató pointerrel tér vissza, ami a benne épített lista első elemére mutat. Paraméterei, a feldolgozásra váró sztring és az a TableList típusú láncolt lista, amiben esetleg már lehetnek elemek. Itt bármelyik paraméter lehet nulla, mert akkor a függvény NULL pointerrel tér vissza.

- **char * makeStrFromTables (TableList* first_elemnt):** Ez a függvény visszacsinálja, amit a processTables függvény csinált. Erre a fájlba íráskor van szükség. Tehát a paraméterben kapott listát sztringgé alakítja, majd visszaadja azt. Lehet NULL, ekkor a sztring is üres lesz.
- **TableList* removeTableElement (TableList* first, int const code):** Ez a függvény eltávolítja a paraméterben megadott konstans egész szám értékével megegyező asztalszámú listaelemet, majd visszatér a lista első elemére mutató pointerrel. A lista TableList típusú, és a code-dal együtt lehet nulla, mert olyankor nem történi semmi.
- **bool isExist (TableList * first_element, int const table_number):** A paraméterben kapott TableList típusú lista és a konstans egész asztal kód alapján visszatér logikai igazzal vagy hamissal, ha van olyan listaelem, aminek a kódja megegyezik a megadott kóddal.
- **TableList * findTableByCode (TableList* table_list, int const code):** A paraméterben kapott TableList típusú lista és a konstans egész asztal kód alapján visszatér annak az elemnek a helyével, ahol annak a kódja megegyezik a megadott kóddal. Ha nincs ilyen elem, akkor NULL-al tér vissza.
- **void freeUpTables (TableList* first):** Felszabadítja a paraméterben kapott TableList típusú láncolt listát. Visszatérési értéke nincs.
- **TableList* manageTablesMP (TableList * first_element):** Az 1. menüpont almenüjének megjelenítésére és működtetésére szolgáló függvény. Kitudja írni a paraméterben kapott TableList típusú láncolt lista elemeit, bővíteni tudja azt, valamint törölni tud elemeket. Kilépésnél visszatér a módosított lista első elemére mutató pointerrel. Módosítja a tables.txt fájlt, minden listaváltozásnál.
- **TableList * setIsFree (TableList* first,int const wich_table, bool const value):** Megkeresi és beállítja a paraméterként kapott adatok alapján a lista azon elemének foglaltságát jelző változó értékét, ahol az asztal kódja megegyezik a megadott kóddal. Majd visszatér a módosított listával.
- **TableList * setReservation (TableList * first):** A 3. menüpontot megvalósító függvény. Bekéri, majd módosítja az asztal foglaltságát a megfelelő függvények felhasználásával. Ezután módosítja a tables.txt fájlt.

timemanage

- **typedef struct Time..:** Ez egy olyan struktúra, ami egy időpont tárolására alkalmas. Tartalmaz egy egész változót, ami a napokat tárolja 0-6, vasárnaptól szombatig. Egy logikai változót, ami különböző igaz-hamis értékeket jelezhet. Jelen programban nyitva tartás esetén azt jelzi, hogy az étterem aznap nyitva van-e, előrefoglalás esetén pedig azt, hogy a vendég megérkezett-e már. A kezdő-óra,-perc és a vég-óra,-perc értékeket is egész változók tárolják a struktúrán belül.
- **void processOH (char * base_str, Time *final_OH, int const n):** A paraméterben kapott sztring és konstans egész tömb elemeinek száma segítségével feltölt egy n nagyságú tömböt értékekkel. Majd ezt a cím szerint kapott struktúra típusú tömbben adja vissza. A bemenő tömbnek léteznie kell előtte, és a hosszának meg kell egyeznie n-nel. Egyébként nincs visszatérési értéke.
- **bool isOpenNow ():** Megadja, hogy az étterem a beolvasott openinghours.txt alapján éppen nyitva van-e. Ez alapján tér vissza logikai igaz vagy hamis értékkel. Ha nincs bemeneti nyitva tartás fájl, akkor bekéri azt, addig nem engedi használni a programot.

Dátum: Budapest, 2019.11.30

Benedek Katalin

Benedek Katalin