Programozói dokumentáció

Tartalom

1 Étterem adminisztráció

A program célja egy étterem adminisztrációja ebbe bele értődik a rendelések felvétele, számlázás, menü rögzítése, új asztalok nyitása és egy foglaltsági térkép. Az írt program ezeket mind megvalósítsa és fájlokban menti az adatokat, hogy újraindítás során ne vesszenek el. Az éttermet három helység képezi: terasz, földszint és emelet, minden helységnek külön txt típusú fájlja van amiben minden sorban el van tárolva egy asztal férőhelyeinek a száma és egy szóközzel elválasztva az asztal számlája. Ezekben a helységekben találhatók asztalok, esély van új asztalok elhelyezésére, valamint asztalok elvitelére. A programot `switch` menük vezérlik. A modulok azokat a függvényeket tartalmazzák, amik a source fájlok nevében leírt feladat végrehajtásában vesznek részt.

2 Adatszerkezetek

A programban két fő adatszerkezet van használva. Az egyik a menü adatait tároló struktúra, ami tartalmazza egy terméknek a nevét, valamint az árát és láncolt listában tárolja őket. Az adatokat egy `menu.txt` nevű fájlból nyeri. A fájlban mindegyik sorban szerepel egy termék neve valamint szóközzel elválasztva a termék neve. Egy termék adatai tároló struktúra:

```
typedef struct menuAdat{
    char termek[51];
    int ar;
    struct menuAdat *kov;
}menuAdat;
```

Az asztalok adatai tároló struktúra tárolja minden asztal férőhelyeinek számát és a számláját, mivel fizikailag nem lehet egy helységben olyan sok asztal, ami drasztikusan befolyásolná a program teljesítményét ezért az említett adatok egy dinamikus tömbbe vannak tárolva. Mivel az asztalok sorszáma 1-től kezdődik és egyesével növekedik, ezért nem tároltam őket, mert a sorszámok valójában a tömb indexei, amik fájlból olvasáskor kapnak értéket. Egy asztal adatait tároló struktúra:

```
typedef struct asztalAdat{
    int szamla;
    int ferohely;
}asztalAdat;
```

3 A program működését vezérlő fő függvények

`menuAdat* beolvasMenu()`

Fájlból beolvassa láncolt listába a menü összes termékét és az visszatérít egy pointert, ami az első adatra mutat. Ha nem volt sikeres olvasás akkor NULL pointert térít vissza. Nem szükséges paramétereket átadni mert láncolt lista feltöltése az nélkül is megvalósítható és a termékek adatai csak a `menu.txt` fájlba találhatóak.

`void asztalBeolvas(asztalAdat *asztal, char const *fajlnev)`

Fájlból beolvassa egy dinamikusan foglalt tömbbe egy adott helységben lévő asztalok adatait. A függvénynek nincs visszatérítési értéke, mert a tömbnek a memória a sorok számának megfelelően van foglalva, ha az adott fájlban nincs egy sornyi adat se akkor a függvény ahol használva van az `asztalBeolvas` leáll.

4 Rendelés felvételéhez kapcsolódó lényeges függvények

`int termekAr(char const *termek)`

Ez a függvény visszatéríti egy paraméterként átvett terméknek az árát, ha nem talált olyan terméket NULL-ot térít vissza.

```
void rendelesFelvetelFajl(char const *fajlnev)
```

Paraméterként átvett fájlba, a függvényben beolvasott sorszámú asztalhoz hozzáadja a szintén billentyűzetről beolvasott termék árát.

```
void rendelesFelvetelSegito(asztalAdat *asztal,char const *fajlnev,int const n)
```

Parméterként átvett fájlba beírja a szintént paraméterként átvett `asztalAdat` tömb n-darab adatát. Mindegyik sorba egy asztal férőhelyét és a számláját.

5 Számlázáshoz kapcsolódó lényeges függvényei

```
int szamlanyomtatasMegerosites()
```

A függvényben lévő `switch` 1-et térít vissza `case 1`-re, más esetre 2-őt.

```
void szamlaNyomtatasFajl(char const *fajlnev)
```

Billentyűzetről beolvasott sorszámú asztal számláját 0-ra állítja. ha fenti függvény egyet térített vissza, másképp nem változtat a számlán. Jelzi, ha a számla már az adatok beolvasásnál 0.

6 Menü rögzítése lényeges függvényei

```
void menuTorol(menuAdat **menu,char const *etel)
```

Töröli azt az elemet a láncolt listából amelyiknek az adata `strcmp` szerint megegyezik a paraméterként átvett stringel. Ha üres a lista hibaüzenetet ír ki a képernyőre és leáll a függvény.

```
void menuFajlbaIr(menuAdat *menu)
```

Ha a láncolt lista nem üres, tartalmát beírja soronként `menu.txt` fájlba. A sorok tartalmazzák egy termék árát és szóközzel leválasztva a termék nevét.

```
void menuPrint()
```

Kiírja a `menu.txt` fájl tartalmát. Soronként egy termék nevét és árát írja ki.

```
void felszabadit(menuAdat *menu)
```

Felszabadítja a láncolt listát.

7 Asztal kezeléshez tartozó függvények

```
void asztalFelFajlba(char const *fajlnev)
```

Egy új sort hozzáad a fájlhoz (új asztal felvétele), aminek nevét paraméterként veszi át a függvény. Billentyűzetről beolvassa, hogy hány férőhelyes az asztal a számlát 0-nak tekinti.

```
void asztalTorolFajlba(char const *fajlnev)
```

Adatokat olvas be egy tömbbe, fájlból aminek nevét paraméterként veszi át, majd billentyűzetről beolvasott indexen kívül a tömb összes elemét visszaírja a fájlba (asztal törlés). A törlés azért történik így mert hatékonyabb, mint törölni egy elemet egy tömböl majd az beírni a fájlba.

```
void asztalMenu()
```

`switch` menü az elvégezni kívánt feladat kiválasztásához

8 Foglaltsági térkép függvénye

```
void uresAsztalok(char *fajlnev)
```

Kiírja a képernyőre azoknak az asztaloknak a sorszámát és férőhelyét ahol a számla 0.