|  |  |
| --- | --- |
| **Tantárgy** | A programozás alapjai 1. |
| **Feladat** | Étterem |

Programozói dokumentáció

Modulok

* **menu:** Leírja a menü tárolására alkalmas adatstruktúrákat, és definiálja a menü memóriakezelő és fájlkezelő függvényeit.
* **rendelesek:** Leírja a rendelések tárolására alkalmas adatstruktúrákat, és definiálja a rendelések memóriakezelő függvényeit.
* **asztalok:** Leírja az asztalok tárolására alkalmas adatstruktúrákat, és definiálja az asztalok memóriakezelő és fájlkezelő függvényeit. (a rendelések fájlkezelő függvényei is ide tartoznak)
* **funkciok:** Definiálja az adatstruktúrák kezelésére alkalmas függvényeket. (nagyrészt a specifikációban kifejtett funkciók megvalósítása)

Adatstruktúrák

A menü adatainak eltárolása

|  |  |
| --- | --- |
| **struct Menupont** | Egy menüpont adatait eltároló *struktúra* |
| int azonosito | A menüpont azonosítója (a fájlban definiált sorrend szerint 0-tól kezdve) – *egész szám* |
| char nev[50 + 1] | A menüpont neve (maximum 50 karakter + lezáró nulla)  - *szöveg* |
| int ar | A menüpont ára – *egész szám* |
| struct Menupont \*kov | A következő menüpont címe a láncolt listához - *mutató* |

|  |  |
| --- | --- |
| **struct Menu** | A menü láncolt listát eltároló *struktúra* |
| struct Menupont \*eleje | Az első menüpont címe a listában - *mutató* |
| struct Menupont \*vege | Az utolsó menüpont címe a listában- *mutató* |

Azért a láncolt listára esett a választás ehhez a problémához, mert nem tudjuk előre, hogy hány elem van a fájlban. Továbbá ezzel a szerkezettel gyorsan be lehet szúrni elemet a lista végére, és egy tetszőleges elem törlése is gyors.

Egy helyhez tartozó rendelések eltárolása

|  |  |
| --- | --- |
| **struct Rendeles** | Egy rendelés adatait eltároló *struktúra* |
| Menupont \*termek | A rendelt termék címe – *mutató menüpontra* |
| int darab | Hány darab terméket rendeltek – *egész szám* |
| struct Rendeles \*kov | A következő rendelés címe - *mutató* |

|  |  |
| --- | --- |
| **struct Rendelesek** | Egy rendelések láncolt listát eltároló *struktúra* |
| struct Rendeles \*eleje | Az első rendelés címe a listában - *mutató* |
| struct Rendeles \*vege | Az utolsó rendelés címe a listában- *mutató* |

Azért a láncolt listára esett a választás ehhez a problémához, mert egy adott asztal adott helyéhez tartozó rendelések száma gyakran változik, ezért gyors beszúrást, törlést kell lehetővé tenni.

Az asztalok adatainak eltárolása

|  |  |
| --- | --- |
| **struct Pozicio** | Egy asztal pozícióját eltároló *struktúra* |
| int X | Az asztal pozíciója az X tengelyen – *egész szám* |
| int Y | Az asztal pozíciója az Y tengelyen – *egész szám* |

|  |  |
| --- | --- |
| **enum Statusz** | Egy asztal státuszát kifejező *enum* |
| SZABAD (0) | Azt jelenti, hogy az asztal nincs megnyitva, nem ül ott senki |
| FOGLALT (1) | Azt jelenti, hogy az asztal éppen meg van nyitva, ülnek ott vendégek. |

|  |  |
| --- | --- |
| **struct Asztal** | Egy asztal adatait eltároló *struktúra* |
| int azonosito | Az asztal azonosítója (a fájlban definiált sorrend szerint 0-tól kezdve) – *egész szám* |
| struct Pozicio pozicio | Az asztal elhelyezkedése az éttermen belül – *pozíció struktúra* |
| enum Statusz statusz | Az asztal státusza (szabad / foglalt) – *státusz enum* |
| int ferohely | Az asztal helyeinek száma (hány szék van) – *egész szám* |
| Rendelesek \*hely\_rendelesek | Az asztal helyeihez tartozó rendelések (minden helyhez egy láncolt lista) - *tömb* |
| struct Asztal \*kov | A következő asztal címe - *mutató* |

|  |  |
| --- | --- |
| **struct Asztalok** | Az asztalok láncolt listát eltároló *struktúra* |
| struct Asztal \*eleje | Az első asztal címe a listában - *mutató* |
| struct Asztal \*vege | Az utolsó asztal címe a listában - *mutató* |

Azért a láncolt listára esett a választás ehhez a problémához, mert nem tudjuk előre, hogy hány elem van a fájlban. Továbbá ezzel a szerkezettel gyorsan be lehet szúrni elemet a lista végére, és egy tetszőleges elem törlése is gyors.

*Egy asztal és a hely\_rendelések 3 hellyel:*

|  |  |  |
| --- | --- | --- |
| rendelés… | rendelés… | rendelés… |
| rendelések láncolt lista | rendelések láncolt lista | rendelések láncolt lista |
| asztal | | |

Függvények

Menü modulban

* **menupont\_foglal:** Lefoglal egy menüpontot.

Bemenet(ek):

1. int azonosito – A lefoglalandó menüpont azonosítója
2. int ar – A lefoglalandó menüpont ára
3. char \*nev – A lefoglalandó menüpont nevét tartalmazó sztring

Kimenet: Az újonnan lefoglalt menüpontra mutató pointer, vagy NULL, ha nem sikerült a foglalás

* **menu\_felszabadit:** Felszabadít egy menüt.

Bemenet(ek):

1. const Menu \*menu – A felszabadítandó menüre mutató pointer

* **menu\_beolvas:** A megadott fájl tartalmát egy menüvé alakítja.

Bemenet(ek):

1. char \*fajl – A fájl útvonalát tartalmazó sztring
2. Menu \*menu – Egy menüre mutató pointer, ebbe írja az adatokat

Kimenet: Visszatérési értéke 0, ha sikerült a beolvasás. 1, ha nem sikerült megnyitni a fájlt. 2, ha nem sikerült hozzáadni valamelyik elemet a listához.

* **menu\_kiir:** Kiírja a menüt a megadott fájlba.

Bemenet(ek):

1. char \*fajl – A fájl útvonalát tartalmazó sztring
2. const Menu \*menu – A kiírandó menüre mutató pointer

Kimenet: Visszatérési értéke 0, ha sikerült a kiírás. 1, ha nem sikerült megnyitni a fájlt.

* **menu\_sor\_hozzaad:** A bemeneti sztringet átalakítja egy menüponttá, és hozzáadja a menühöz.

Bemenet(ek):

1. char \*sor – A menüpont adatait tartalmazó sztring
2. Menu \*menu – A menüre listára mutató pointer, ebbe szúrja be a menüpontot

Kimenet: Visszatérési értéke 0, ha sikerült hozzáadni a menüpontot. 1, ha nem sikerült lefoglalni a memóriát.

Rendelések modulban

* **rendeles\_foglal:** Lefoglal egy rendelést.

Bemenet(ek):

1. Menupont \*termek – A rendelt termék menüpontjára mutató pointer
2. darab – Ahány darabot rendeltek a termékből

Kimenet: Az újonnan lefoglalt rendelésre mutató pointer, vagy NULL, ha nem sikerült a foglalás.

* **rendelesek\_felszabadit:** Felszabadítja a megadott rendelések listát.

Bemenet(ek):

1. const Rendelesek \*rendelesek – A rendelések listára mutató pointer

Asztalok modulban

* **asztal\_foglal:** Lefoglal egy asztalt, az egyes helyekhez tartozó rendeléseket üresre állítja.

Bemenet(ek):

1. int azonosito – A lefoglalandó asztal azonosítója
2. struct Pozicio pozicio – A lefoglalandó asztal pozíciója az étteremben
3. enum Statusz statusz – Az lefoglalandó asztal státusza (szabad / foglalt)
4. int ferohely – A lefoglalandó asztal helyeinek száma

Kimenet: Visszatérési értéke a lefoglalt asztalra mutató pointer, vagy NULL, ha nem sikerült a foglalás.

* **asztalok\_felszabadit:** Felszabadítja a megadott asztalok listát.

Bemenet(ek):

1. const Asztalok \*asztalok – Az asztalok listára mutató pointer

* **asztalok\_beolvas:** A megadott fájl tartalmát asztalok listává alakítja.

Bemenet(ek):

1. char \*fajl – A fájl útvonalát tartalmazó sztring
2. Asztalok \*asztalok – Az asztalok listára mutató pointer, ebbe írja az adatokat

Kimenet: Visszatérési értéke 0, ha sikerült a beolvasás. 1, ha nem sikerült megnyitni a fájlt. 2, ha nem sikerült hozzáadni valamelyik elemet a listához.

* **asztalok\_kiir:** Kiírja az asztalok listát a megadott fájlba.

Bemenet(ek):

1. char \*fajl – A fájl útvonalát tartalmazó sztring
2. const Asztalok \*asztalok – Az asztalok listára mutató pointer

Kimenet: Visszatérési értéke 0, ha sikerült a kiírás. 1, ha nem sikerült megnyitni a fájlt.

* **rendelesek\_beolvas:** A fájl soraiban definiált rendeléseket beírja a megfelelő asztal megfelelő helyének listájába.

Bemenet(ek):

1. char \*fajl – A fájl útvonalát tartalmazó sztring
2. const Menu \*menu – A menü listára mutató pointer
3. Asztalok \*asztalok – Az asztalok listára mutató pointer, ebbe szúrja be a rendeléseket a megfelelő helyre

Kimenet: Visszatérési értéke 0, ha sikerült a beolvasás. 1, ha nem sikerült megnyitni a fájlt. 2, ha nem sikerült hozzáadni valamelyik elemet a listához.

* **rendelesek\_kiir:** Az asztalok egyes helyeihez tartozó rendeléseket kiírja a megadott fájlba.

Bemenet(ek):

1. char \*fajl – A fájl útvonalát tartalmazó sztring
2. const Asztalok \*asztalok – Az asztalok listára mutató pointer

Kimenet: Visszatérési értéke 0, ha sikerült a kiírás. 1, ha nem sikerült megnyitni a fájlt.

* **asztal\_sor\_hozzaad:** A bemeneti sztringet átalakítja egy asztallá és beszúrja az asztalok listába.

Bemenet(ek):

1. char \*sor – Az asztal adatait tartalmazó sztring
2. Asztalok \*asztalok – Az asztalok listára mutató pointer, ebbe szúrja be az asztalokat

Kimenet: Visszatérési értéke 0, ha sikerült a beszúrás. 1, ha nem sikerült lefoglalni a memóriát.

* **rendeles\_sor\_hozzaad:** A bemeneti sztringet átalakítja egy rendeléssé, és beszúrja a megfelelő asztal megfelelő helyének rendeléseit tartalmazó listába.

Bemenet(ek):

1. char \*sor – A rendelés adatait tartalmazó sztring
2. const Menu \*menu – A menü listára mutató pointer
3. Asztalok \*asztalok – Az asztalok listára mutató pointer, ebbe szúrja be a rendelést a megfelelő helyre

Kimenet: Visszatérési értéke 0, ha sikerült a beszúrás. 1, ha nem sikerült lefoglalni a memóriát.

Funkciok modulban

* **asztal\_hozzaad:** Hozzáad egy asztalt a megadott adatokkal a láncolt listához.

Bemenet(ek):

1. struct Pozicio pozicio – Az asztal elhelyezkedése az éttermen belül.
2. enum Statusz statusz – Az asztal státusza (szabad / foglalt)
3. int ferohely – Az asztal helyeinek száma (hány szék van)
4. Asztalok \*asztalok – Az asztalok listára mutató pointer, ebbe szúrja be az asztalt.

Kimenet: Visszatérési értéke 0, ha sikerült hozzáadni az asztalt. 0, ha nem sikerült lefoglalni a memóriát.

* **asztal\_keres:** Megkeres egy asztalt a megadott listában.

Bemenet(ek):

1. int azonosito – A keresett asztal azonosítója
2. const Asztalok \*asztalok – Az asztalok listára mutató pointer, amelyben keresni akarunk.

Kimenet: Visszatérési értéke a keresett asztalra mutató pointer, vagy NULL, ha nem találta meg a listában.

* **asztal\_megnyit:** Megnyitja az adott asztalt (a státuszát foglaltra állítja)

Bemenet(ek):

1. int azonosito – A megnyitandó asztal azonosítója
2. Asztalok \*asztalok – Az asztalok listára mutató pointer, ebben keresi az asztalt

Kimenet: Visszatérési értéke 0, ha sikerült megnyitni az asztalt. 1, ha az adott azonosítóval nincs asztal definiálva a listában.

* **asztal\_lezar:** Lezárja az adott asztalt (státuszát szabadra állítja, és törli a hozzá tartozó rendeléseket)

Bemenet(ek):

1. int azonosito – A lezárandó asztal azonosítója
2. Asztalok \*asztalok – Az asztalok listára mutató pointer, ebben keresi az asztalt

Kimenet: Visszatérési értéke 0, ha sikerült lezárni az asztalt. 1, ha az adott azonosítóval nem található asztal.

* **asztal\_torol:** Törli az adott asztalt és csökkenti az azt követő asztalok azonosítóját 1-el.

Bemenet(ek):

1. int azonosito – A törölendő asztal azonosítója
2. Asztalok \*asztalok – Az asztalok listára mutató pointer, amelyben törölni akarunk.

Kimenet: Visszatérési értéke 0, ha sikerült a törlés. 1, ha nem található az asztal, vagy foglalt a státusza.

* **menupont\_hozzaad:** Hozzáad egy menüpontot a listához a megadott adatokkal.

Bemenet(ek):

1. int ar – A menüpont ára
2. char \*nev – A menüpont nevét tartalmazó sztring
3. Menu \*menu – A menü listára mutató pointer

Kimenet: Visszatérési értéke 0, ha sikerült a hozzáadás. 1, ha nem sikerült lefoglalni a memóriát.

* **menupont\_keres:** Megkeres egy menüpontot a megadott listában.

Bemenet(ek):

1. int azonosito – A keresett menüpont azonosítója
2. const Menu \*menu – A menü listára mutató pointer, amelyben keresni akarunk.

Kimenet: Visszaadja a keresett menüpont címét, vagy NULL-t, ha nem található a menüpont a megadott listában.

* **menupont\_torol:** Törli az adott menüpontot és csökkenti az azt követő menüpontok azonosítóját 1-el.

Bemenet(ek):

1. int azonosito – A törölendő menüpont azonosítója
2. Menu \*menu – A menü listára mutató pointer
3. const Asztalok \*asztalok – Az asztalok listára mutató pointer

Kimenet: Visszatérési értéke 0, ha sikerült a törlés. 1, ha nem található menüpont a megadott azonosítóval, vagy ha van a megadott azonosítóval nyilvántartott rendelés.

* **rendeles\_hozzaad:** Hozzáad egy rendelést a listához.

Bemenet(ek):

1. int termek\_azonosito – A termék azonosítója, amelyet hozzá szeretnénk adni
2. int darab – Ahány darabot szeretnénk rendelni a termékből
3. const Menu \*menu – A menü listára mutató pointer
4. Rendelesek \*rendelesek – A rendelések listára mutató pointer

Kimenet: Visszatérési értéke 0, ha sikerült hozzáadni a rendelést. 1, ha nem sikerült lefoglalni a memóriát.

* **rendeles\_menuponttal:** Megmondja, hogy van-e nyilvántartott rendelés az adott menüponttal.

Bemenet(ek):

1. int azonosito – A menüpont azonosítója
2. const Asztalok \*asztalok – Az asztalok listára mutató

Kimenet: Visszatérési értéke 1, ha van rendelés az adott azonosítóval. 0, ha nincs.

[Programozói dokumentáció 1](#_Toc87724388)

[Modulok 1](#_Toc87724389)

[Adatstruktúrák 1](#_Toc87724390)

[A menü adatainak eltárolása 1](#_Toc87724391)

[Egy helyhez tartozó rendelések eltárolása 1](#_Toc87724392)

[Az asztalok adatainak eltárolása 2](#_Toc87724393)

[Függvények 3](#_Toc87724394)

[Menü modulban 3](#_Toc87724395)

[Rendelések modulban 4](#_Toc87724396)

[Asztalok modulban 4](#_Toc87724397)

[Funkciok modulban 6](#_Toc87724398)