**Nagy házi programozói dokumentáció**

**Hotelszobák**

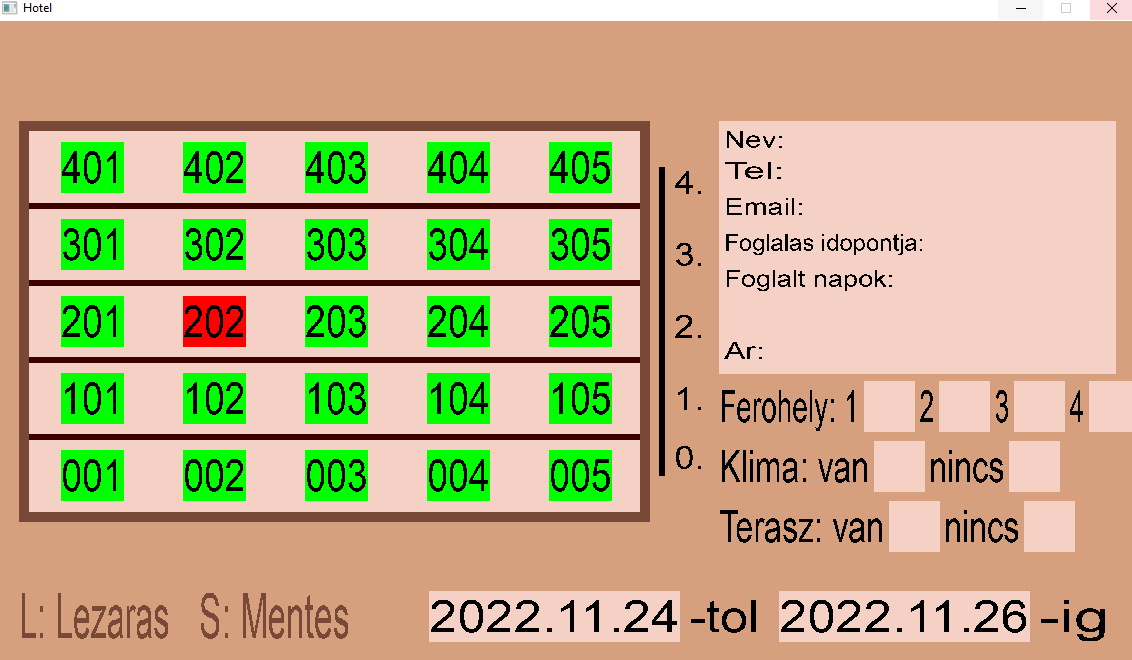
**A kész program**

A program elkészült. Mindkét menü működik. A login ablakban első elindításnál lehet regisztrálni, ez létrehozza a szükséges txt, és bin fájlokat, a többi indításnál, a megadott belépési adatokkal lehet bejelentkezni. A felhasználói oldal is teljesen funkcionális, a -tól -ig dátum megadása után, a program kilistázza a szabad, valamint a dátummal fedésben lévő foglalt helyeket. A program mindig figyeli a dátumok helyes formátumát, valamint az adatmezőknél a maximális hosszúságot, így a program által nem kerülhet érvénytelen adat a szövegfájlokba. Az új felvétel gombbal tudunk új vendégeket fölvenni, a törléssel tudjuk törölni őket. Az L billentyűvel újbóli beléptetésre kényszerítjük a programot, s billentyűvel mentünk.

A Login ablak futásidőben:



A Felhasználói ablak futásidőben, szűrővel beállítva:



**A program felépítése és modulok**

A programban 6 modul van, és ezekhez a modulokhoz 7 header fájl tartozik.

**A modulok**

* main.c: Ez tartalmazza a program fő loopját, ez a loop addig megy, amíg be nem zárjuk a programot. Ez a modul először inicializálja az SDL ablakot, majd bekerül a fő loopba, ahol felváltva inicializálja a belépési oldal, vagy a felhasználói oldal kezdőértékeit attól függően, hogy a felhasználó épp melyik oldalra lép be, majd meghívja a ProgramEventek függvényt, ami a felhasználói eventeket nézi. A modul ezenkívül tartalmazza azokat a változókat, amelyek kellenek mindkét ablakhoz, valamint amelyek a loopot működtetik.
* eventek.c: Ez a modul, a felhasználói eventeket figyeli, olyanokat mint az egérkattintás, backspace, valamint a szövegbevitel. A modul működése abból áll, hogy a ProgramEventek loop figyeli, hogy milyen bemenet érkezett a felhasználótól, s ha az a bemenet érvényes, akkor meghívja az éppen aktív oldal azon függvényét, amely tartalmazza, hogy az egyes gomboknál. beviteli mezőknél melyik függvényt kell meghívni. Ezenfelül itt kerül ellenőrzésre a fájlbeolvasás helyessége, ezért a program csak is akkor léphet tovább a felhasználói oldalra, ha sikerült minden fájlt beolvasnia.
* megjelenites.c: Ez a modul tartalmazza az SDL ablak inicializálás függvényt, valamint minden olyan függvényt, ami megjelenítési parancsokat tartalmaz. Fontos megemlíteni, hogy itt található az a függvény is, amely felhasználói érvényes event hatására képernyőfrissítést végez. Az egyes oldalalak megjelenítendő objektumainak alapértelmezett értékei is itt tárolódnak, például a koordináták, színek, a megjelenítendő szövegek.
* fajladatkezeles.c: Ebben a modulban találhatóak a fájlkezelési függvények, a fájlba írás, olvasás, és az ezekhez tartozó segédfüggvények. Az adatkezelési függvények is itt vannak, a program innen hívja meg a láncolt lista beszúrás, törlés, felszabadítás függvényeket.
* felhasznalomenu.c: Ez a modul határozza meg a felhasználói oldal működését, minden gombhoz, és beviteli mezőhöz létezik egy függvény, ami a szükséges parancsokat tartalmazza. Az eventeket figyelő modul hívja meg ebből a modulból azt a függvényt, ami a kiválasztott gombhoz tartozik.
* bejelentkezesmenu.c: Ez a modul határozza meg a bejelentkezési oldal működését, minden gombhoz, és beviteli mezőhöz létezik egy függvény, ami a szükséges parancsokat tartalmazza. Az eventeket figyelő modul hívja meg ebből a modulból azt a függvényt, ami a kiválasztott gombhoz tartozik.

**Headerek**

A program 7 headert tartalmaz. A menu.c modulon kívül mindegyik modulhoz tartozik egy header fájl, ami az adott modul függvényeit tartalmazza. Ezekben a headerekben sok olyan függvény van, amely egy struktúrával dolgozik, ezért mindegyik header includolja a struktura.h headert is, amely a programban létező összes struktúrát deklarálja.

**Könyvtárak**

Egy SDL program elengedhetetlen építő elemei az SDL könyvtárak. A program ebből hármat használ, az alapvető <SDL.h> könyvtáron kívül használja az <SDL2\_gfxPrimitives.h> könyvtárat, ezzel lehet alakzatokat megjeleníteni a képernyőn, valamint használja a <SDL\_ttf.h> könyvtárat, amivel betűket lehet kirajzolni a képernyőre. Természetesen ezenkívül megtalálhatóak az alapvető könyvtárak is, <string.h> a sztring műveletek miatt, a <math.h> a matematikai műveletek, az <stdbool.h> a bool típusú változók, és az <stdlib.h> a memória műveletek miatt. A memóriakezelés ellenőrzésére szolgál a „debugmalloc.h” külső könyvtár, ezt használja a program, a kiértékelés a futás végén a hiba.txt fájlba íródik.

**Adatszerkezetek és struktúrák**

Az összes struktúra a struktura.h headerben található

1. **typedef** **struct** ar
2. {
3. **int** arak[6];
4. }ar;

Az ár struktúra a hotel áraival töltődik fel, első elindításkor alapértelmezett értékekkel, a többi elindításnál az egysegar.txt fájlból olvassa ki az értékeket. 6 értéket képes tárolni, 0-3 index között a férőhelyek árát, 4. indexben a klíma, 5. indexben a terasz árát tartalmazza.

1. **typedef** **struct** gomb
2. {
3. **int** kezdposX, kezdposY, vegposX, vegposY;
4. **int** nevkezdposX, nevkezdposY, nevvegposX, nevvegposY;
5. **char** nev[31];
6. **bool** aktiv;
7. **bool** kivalasztva;
8. **int** r, g, b;
9. }gomb;
11. **typedef** **struct** alakzat
12. {
13. **int** kezdposX, kezdposY, vegposX, vegposY;
14. **int** r, g, b;
15. }alakzat;
17. **typedef** **struct** szo
18. {
19. **int** kezdposX, kezdposY, hossz, szelesseg;
20. **char** nev[31];
21. **int** r, g, b;
22. }szo;

A fenti három struktúra a három megjelenítendő objektumot deklarálja. A program gombot, alakzatot, vagy szöveget jelenít meg. A fő különbség közöttük funkciójukban van, a gombnak reagálnia kell a felhasználó bemenetre, az alakzat, és szavak csak dizájn elemként vannak jelen, nem reagálnak a felhasználói interakcióra. A gomb struktúra tartalmazza a négyzet alak négy szélének pontját. Mivel a szövegdoboz változhat, ezért annak is külön változókat hoztunk létre. Alapértelmezetten a szövegdoboz akkora, mint maga a gomb. A név karaktertömb a gomb nevét tartalmazza. Ha beviteli mezőként funkciónál a gomb, akkor ez a kartertömb töltődik fel a bemeneti adatokkal. 31 elemű a tömb, maximum 30 karakter hosszú lehet egy adat. Az aktív bool változó azt mondja meg, hogy a gomb kattintható legyen e vagy sem, a kivalasztva változó azt határozza meg, hogy a gomb megjeleníthető e. Az r,g,b int változók a gomb hátterének színét tartalmazzák. Az alakzat struktúra az egyszerű dizájnelemeket jelenti, valamint egy ilyen változóval van megoldva a szobák kijelölése (Egy fekete téglalap). A szavak struktúra az egyszerű szavakat jeleníti meg.

1. **typedef** **struct** hotelszobak
2. {
3. **int** ferohely;
4. **int** klima;
5. **int** terasz;
6. }hotelszobak;

A hotelszobak struktúra tudja eltárolni egy szoba adatait, azt, hogy hány férőhelyes, tartozik e a szobához klíma, illetve terasz. A program egy ilyen struktúrából hoz létre egy 25 elemű tömböt, s feltölti az összes szoba alapértelmezett adatával A klíma, valamint a terasznál 1-es jelöli ha van az adott szolgáltatás, 2-es ha nincs.

1. **typedef** **struct** FelhasznaloAdatok
2. {
3. **char** felhasznalonev[21];
4. **char** jelszo[21];
5. }FelhasznaloAdatok;

A FelhasznaloAdatok struktúrába olvassa be a program a felhasználó által bevitt adatokat, s a struktúrát adja tovább ellenőrzésre a fájlellenőrző függvénynek. Maximum 20 karaktert lehet beírni, ezért 21 eleműek a karaktertömbök.

1. **typedef** **struct** adatoklista
2. {
3. **int** foglalasidatum;
4. **int** tmeret;
5. **struct**  szemelyadatok \*szemelyek;
6. **struct**  adatoklista \*kov;
7. }adatoklista;

Ez a láncolt lista struktúra, amely tartalmazza a foglalási dátumot, a dinamikusan foglalt tömb méretét, a dinamikusan foglalt tömb pointerét, s a következő listaelem pointerét. Azért választottam ezt az adatstruktúrát, mert a foglalasok.txt fájl foglalási dátumonként sorolja egy csoportba a vendégek adatait, ezért a láncolt lista egy elemében a foglalási dátum közös, a vendégek többi adata pedig a dinamikusan foglalt szemelyadatok struktúra tömbbe kerül bele.

1. **typedef** **struct** szemelyadatok
2. {
3. **char** nev[31];
4. **char** tel[31];
5. **char** email[31];
6. **char** szobaszam[4];
7. **int** startdatum;
8. **int** stopdatum;
9. **int** ar;
10. }szemelyadatok;

Ilyen szemelyadatok struktúrában tárolódik a foglalási dátumon kívül a vendégek összes többi adata. A karaktertömbök 31 eleműek, mivel a program által maximum 30 karaktert lehet egy beviteli mezőbe írni, az üdülés kezdete, és vége dátum int változóban eltárolva, mivel így könnyű nagyságrendileg összehasonlítani a dátumokat, az ár változó is int ként van tárolva.

1. **typedef** **struct** ListazandoFoglalasok
2. {
3. adatoklista \*lista[25];
4. **int** foglalasindex[25];
5. }ListazandoFoglalasok;

Ennek a struktúrának a kilistázásnál van lényege. Amikor megadunk egy -tól -ig dátumot, és a program megkeresi, hogy mely foglalásokkal van átfedésben a dátum, akkor a 25 szobából ahová talál foglalást, oda beírja a láncolt listaelem pointerét, a foglalásindex pedig meghatározza, hogy a láncolt listaelemen belüli dinamikus tömbben melyik indexen helyezkednek el a vendég adatai. Ez azért jó, mert bármikor, mikor kiválasztjuk a foglalt szobát, nem kell megkeresnie a hozzátartozó vendéget, mert ez a struktúra egyértelműen azonosítja, s így egyből el lehet érni az adatait, nem kell végigmenni a listán.

**Függvények modulonként**

**main.c**

1. **int** main(**int** argc, **char** \*argv[])

A main modul egyetlen függvényt, a main függvényt tartalmazza. Innen ágazódnak szét a függvényhívások. A main függvény közvetlenül az SDL ablak inicializálás, a login, és felhasználó oldal inicializálás, valamint, a felhasználó eventek függvényt tudja meghívni. Itt van definiálva a láncolt lista, valamint a megjelenítendő objektumok is.

**fajladatkezeles.c**

1. ar alapertelmezettar();

Ez a függvény az ar struktúrát tudja feltölteni az alapértelmezett árakkal. Első programindításnál kerül meghívásra.

1. adatoklista \*Beszuras(adatoklista \*start, **int** foglalasidatum, **int** meret);

A láncolt listaelem beszúrásánál használjuk a függvényt. Paraméterül veszi át a lista elejét, a foglalási dátumot, valamint a listán belüli dinamikusan foglalt tömb méretét. Mivel a beszúrás a lista elejére szúr be elemet, ezért meg fog változni az első elem pointere, ezért a visszatérési érték az új első elem pointer lesz. A beszúráson kívül a függvény a létrehozott elemnek odaadja a foglalási dátumot, valamint, hogy mekkora mérető a személyadatok tömb, vagyis, hogy mennyi vendég adatai vannak a listaelemen belül.

1. adatoklista \*TeljesFelszabadit(adatoklista \*lista);

Ezt a függvényt a program a felhasználói oldal bezárásakor használja. Ez két esetben történik, ha a felhasználó bezárja a programot, vagy ha átvált a login oldalra. A függvény paraméterként veszi át a láncolt lista első elemét, és mivel a végén magát az első elem pointerét is NULL-ra állítja ezért azt adja vissza. A függvényen belül nem csak a láncolt lista memóriaterülete szabadul fel, hanem a listaelemen belüli dinamikus tömb memóriaterülete is.

1. adatoklista \*ReszlegesFelszabadit(adatoklista \*lista, adatoklista \*keresendo);

Ez a függvény csak részleges felszabadítást végez, ilyen akkor van, ha a felhasználó kitörli az utolsó vendég adatait is a listaelemből. Paraméterek a lista első eleme, valamint a törlendő listaelem. Mivel lehetséges, hogy az első elemet kell törölni, ezért a visszatérési érték az első elem pointere.

1. **void** Kereses(gomb \*gombok, ListazandoFoglalasok \*listazandofoglalasok, adatoklista \*lista, **int** \*datumertek);

A -tól -ig dátum beírása után kerül meghívásra, kilistázza a megadott dátummal fedésben lévő foglalásokat. A függvény paraméterül kapja a gombok tömböt, mivel a gombok neve a hotelszoba száma, s azt hasonlítja össze a vendégek szobaszámával. A listazandoparameter tömbbe tölti be a program a láncolt lista pointerét, és a vendég indexét, ha van foglalás. A dátumérték tömbbe kapja meg a függvény, hogy mely két érték között kell nézni, hogy fedésben van e a vendég foglalási dátuma.

1. **bool** FoglalasTorles(ListazandoFoglalasok \*listazandofoglalsok, adatoklista \*\*lista, **int** index);

Ez a függvény akkor kerül meghívásra, ha a felhasználó törölni akar egy foglalást. Paraméterül kapja a listazandofoglalasok tömböt, ez azért szükséges, mert ez már egyből megjelöli melyik vendéget kell törölni, megkapja a láncolt listát pointer pointerként, mivel lehetséges, hogy változtatni kell rajta, s megkapja a listazandofoglalasok megfelelő indexét. A függvény különlegessége, hogy ne kelljen a láncolt listán belüli dinamikus tömböt minden egyes törlésnél újraméretezni, ezért csak akkor méretezi újra a tömböt, ha 10 üres tömbelem van. Ezt úgy éri el, hogy minden egyes törlésnél, az új törléssel együtt megnézi, hogy megvan e a 10 üres tömbbelem, s akkor újraméretez, máskülönben csak töröl. A törlés azt jelenti, hogy beállítja az inaktív értéket, vagyis a startdatum, stopdatum -1 es értékét. A visszatérési érték egy bool változó, azt adja meg, hogy sikeres volt e törlés.

1. **void** ToroltErtekBeallitas(ListazandoFoglalasok \*listazandofoglalsok, **int** index);

A segédfüggvény feladata törlésnél a kijelölt vendég adatának invalidra állítása, vagyis a startdátum, és a stopdátum -1 -es értékre állítása. Mindezek után a tömbbelem hátramozgatása az invalid értékekhez. Paraméterként a listazandofoglalasok tömböt veszi át, ezért tudja egyből elérni a vendég adatait, a másik paraméter a listazandofoglalsok tömb indexe.

1. **bool** TorlesUjraMeretezes(ListazandoFoglalasok \*listazandofoglalsok, **int** index);

Segédfüggvény, igazat ad ha az aktuális törléssel együtt 10 nem használt tömbelem keletkezik. A paraméterei a listazandofoglalasok, és az index, ezek segítségével egyből el tudja érni a vizsgálandó tömböt.

1. **bool** FoglalasFelvetel(ListazandoFoglalasok \*listazandofoglalasok, adatoklista \*\*lista, **int** index, **char** adatok[][31], **int** datumok[3], **int** ar);

Ez a függvény, akkor kerül meghívásra, ha a felhasználó foglalni akar. Paraméterként kapja a listazandofoglalasok tömböt, azért, hogy odaadja a foglalás információit, vagyis a láncol lista pointerét, valamint az ügyfél indexét a listaelemen belüli dinamikus tömbben. A láncol lista paraméter pointer pointer, mivel lehetséges, hogy a foglalás során megváltozik a kezdő pointer értéke. Az index azt jelöli, hogy a listazandofoglalasok tömbben melyik helyre íródjon az adat, az adatok tömb tartalmazza a beírt foglalási adatokat, a datumok, a foglalasi datumot, a startdatumot, és a stopdatumot. Az ár a szoba árát tartalmazza. A függvény, ha az aktuális foglalással betelt a dinamikus tömb, akkor úgy méretezi át azt, hogy ráhagy +5 helyet, azért, hogy ne kelljen mindig átmásolni az egész tömböt. Ha teljesen új foglalási dátumot adtunk meg, létrehoz egy új listaelemet.

1. **void** FoglalasBeiras(adatoklista \*kivalasztottelem, **int** index, **char** adatok[][31], **int** datumok[3], **int** teljesar);

Foglalás estén ez a függvény írja be a foglalás adatait a láncol listába. Paraméterként veszi át a láncolt lista kiválasztott elemét, a listán belüli dinamikus tömb indexét, és az adatokat.

1. adatoklista \*FoglalasListaLetrehozas(adatoklista \*lista, **int** datum);

Ez a függvény foglalásnál, ha a láncolt listában megtalálja a megadott foglalási dátumot akkor visszaadja annak pointerét, máskülönben NULL pointert ad, ezáltal megkapjuk vagy azt a helyet ahová írni kell az adatokat, vagy NULL pointert kapunk ami jelzi, hogy nem talált a listában ilyen elemet.

1. **bool** FoglalasUjraMeretezes(adatoklista \*kivalasztottelem);

Ez a függvény igazad ad vissza ha az aktuális foglalásnak van hely, hamisat ha bővíteni kell a dinamikus tömböt. Paraméterül azt a listaelemet kapja amelyik azt a tömböt tartalmazza amelyet ellenőrízni kell.

1. **int** ElemIndex(adatoklista \*kivalasztottelem);

Megtalálja a legelső memória helyet ahova beírható a vendég adata. Paramétere az a láncolt listaelem ahol az ellenőrízni kívánt dinamikus tömb található, visszatérési értéke a szabad hely indexe.

1. **void** DatumvaltasVissza(**int** datum, **char** \*atvaltas);

Segédfüggvény, az int típusú dátumot alakítja vissza karaktertömbbé. Paraméterül veszi át a dátumot, és egy karaktertömböt, ahova az átalakítást végzi.

1. **int** Datumatvaltas(**char** \*datum, **bool** datumfajta);

Dátumátváltás, karaktertömb dátumot vált át int típusúvá, a visszatérési értéke a dátum.

1. **bool** FajlLetezik(**char** \*fajlnev);

Igazat ad vissza ha a paraméterben megadott fájl létezik. Azt ellenőrzi a program vele, hogy első elindítás vagy sem. Ha létezik a bejelentkezési adatokat tartalmazó bináris fájl akkor többszöri indítás.

1. **bool** FajlbaIras(**char** \*fajlnev, **int** meret, **char** adatok[][31]);

Soronként ír a fájlba. Paraméterül a fájlnevet, a sorok számát, valamint az adatokat kapja.

1. **bool** FajbolBeolvasas(**char** \*fajlnev, **int** meret, **char** adatok[][31]);

Ez a függvény olvassa be az árakat, és a bináris fájlt. Paraméterül a fájlnevet, a sorok számát, valamint a tömböt kapja ahova az adatokat be kell olvasni. Sikeres beolvasás esetén igazat ad vissza.

1. **bool** FoglalasBeolvas(**char** \*fajlnev, adatoklista \*\*lista);

A foglalásokat olvassa be, paraméterül a fájlnevet kapja, valamint a láncolt lista kezdő elemét pointer pointerként, mivel itt kap értéket a láncolt lista. A függvény beolvas egy sort, majd megnézi, hogy „#” -el kezdődik e, illetve, hogy a hossza nagyobb e mint 1. Ha nagyobb a hossza mint 1 akkor biztosan a foglalási dátum van abban a sorban. Ekkor megszámolja, hogy mennyi sor tartozik ahhoz a foglalási dátumhoz, majd beszúr egy láncolt lista elemet, s lefoglalja benne a vendégek dinamikus tömbjét akkora mérettel mint amennyi sort számolt. Ezután feltölti a dinamikus tömböt az adatokkal. Ezután tovább keresi a következő foglalási dátumot. Fájl vége jelig megy, és igazad ad vissza ha sikerült a beolvasás.

1. **int** Sorosszeg(**FILE** \*fajl);

Ez a függvény számolja meg hogy a foglalási dátumhoz hány sor vendég adat tartozik. Paraméterként a fájlt kapja, visszatérési értéke a sorok száma.

1. **void** AdatInicializalas(szemelyadatok \*foglalas, **int** meret, **int** kezdoertek);

Ez a függvény tölti fel invalid értékekkel a vendégek adatait tartalmazó tömböt. Ahol van valós érték ott ez felül lesz írva. Paraméterei a vendégek dinamikus tömbje, a tömb mérete, valamint a kezdőérték, hogy honnan kezdje a feltöltést.

1. **void** Adatkitoltes(szemelyadatok \*foglalas, **char** \*sor, **int** index);

Kitölti a dinamikus tömb indexnél lévő elemét a beolvasott sor értékeivel.

1. **void** FoglalasIras(**char** \*fajlnev, adatoklista \*lista);

Végigmegy a láncolt listán, formázza az adatokat, majd soronként fájlba írja őket. Paraméterként kapja a fájl nevét, valamint a láncolt lista kezdőelemét.

**megjelenites.c**

1. **void** AblakLetrehozas(SDL\_Renderer \*\*renderer, SDL\_Window \*\*window);

Létrehozza a SDL renderert, és az SDL ablakot.

1. **void** GombhozRendeles(gomb \*gombok, **int** meret, **int** koordinatak[][4], **char** nevek[][31], **int** szinek[][3], **bool** \*aktivak, **bool** \*kivalasztottak);

Hozzárendeli a megadott gombok tömbhöz a másik tömbökben megadott értékeket. Az oldalak létrehozásánál hívódik meg. Paraméterül a gomb tömböt, a tömb méretét, illetve a gombokhoz hozzárendelni szükséges adatokat kapja.

1. **void** SzohozRendeles(szo \*szavak, **int** meret, **int** koordinatak[][4], **char** nevek[][31], **int** szinek[][3]);

Ugyanaz mint a gomboknál, csak most a szó tömbhöz rendel hozzá.

1. **void** AlakzathozRendeles(alakzat \*alakzatok, **int** meret, **int** koordinatak[][4], **int** szinek[][3]);

Ugyanaz mint az előző kettő, csak alakzatokkal.

1. **void** LogOldal(SDL\_Renderer \*renderer, **char** \*letezik, gomb \*gombok, szo \*szavak);

Ez a függvény tartalmazza a belépés oldal minden megjelenítendő objektumának az értékeit. Meghívásra az előző függvények segítségével hozzárendeli a tömbökhöz az értékeket, s lefrissíti először a képernyőt. Paraméterül a képernyőfrissítéshez a renderert kapja. A letezik pointer azt jelöli, hogy a login oldalon a belépés vagy a regisztráció szót kell megjeleníteni. A gombok, és szavak a megjelenítendő objektumok tömbje.

1. **void** FelhasznaloOldal(SDL\_Renderer \*renderer, gomb \*gombok, alakzat \*alakzatok, szo \*szavak);

Ugyan az mind az előző függvény, csak a felhasználó oldal értékeit tartalmazza.

1. **void** SzoMegjelenites(SDL\_Renderer \*renderer, szo \*szavak, **int** szomeret, gomb \*gombszo);

Ez a függvény felel azért, hogy megjelenítse a megadott szavakat a képernyőn. Ha gomb szavát adjuk át neki, akkor egyszer végzi el a parancsokat, ha szavak tömböt adunk neki akkor a szomeretig loopol, s kiadja az összes szónak a megjelenítési parancsot. Azért szükséges az esetszétválasztás, mert ez a függvény több helyen van használva.

1. **void** TeglalapMegjelenites(SDL\_Renderer \*renderer, alakzat \*alakzatok, **int** alakzatmeret, gomb \*gombhatter);

Téglalapot jelenít meg. Itt is esetszétválasztás történik, ugyanis használja a program alakzatok, valamint gomb hátterének kirajzolásához is.

1. **void** Gombmegjelenites(SDL\_Renderer \*renderer ,gomb \*gombok, **int** meret);

Ez a függvény ötvözi az előző két függvényt, és így jeleníti meg a gombokat. Paraméterei az SDL renderer, a gombok tömb, és a tömb méret. Végig loopol a gombokon, s csak akkor adja ki a megjelenítési parancsokat, ha engedélyezve van a gomb megjelenítése.

1. **void** GombTiltasFeloldas(gomb \*gombok, **int** meret, **int** kezdoertek, **bool** valaszt);

Ez a függvény feloldja vagy letiltja a gombok kattinthatóságát. Paraméterül a gombok tömböt, a méretet, a kezdő értéket, hogy honnan kezdje a loopot, valamint a bool kapja ami megmondja, hogy engedélyezés legyen vagy tiltás.

1. **void** GombMegjelenitesTiltasFeloldas(gomb \*gombok, **int** meret, **int** kezdoertek, **bool** valaszt);

Ugyanaz mint a felső függvény csak a gomb megjelenítését lehet engedélyezni vagy tiltani.

1. **void** Kepernyotorles(SDL\_Renderer \*renderer);

Törli a teljes képernyőt.

1. **void** FoglaltsagStatuszMegjelenites(gomb \*gombok, ListazandoFoglalasok \*listazandofoglalasok);

Ez a függvény felel azért, hogy meg legyen különböztetve a foglalt, illetve szabad szóba. Végigloopol a listazandofoglalasok tömbön, s ahol NULL pointer van az szabad tehát zöldre színezi a gombot, ami nem NULL az foglalt tehát pirosra színezi a gombot.

1. **void** Kepernyofrissites(SDL\_Renderer \*renderer, gomb \*gombok, **int** gombmeret, alakzat \*alakzatok, **int** alakzatmeret, szo \*szavak, **int** szomeret);

Képernyőfrissítést végez, itt lép érvénybe a sok kiadott kirajzolási parancs. Paraméterül kapja a három megjeleníthető objektum tömbjét és a méreteket. Felhasználó event után hívódik meg ez a függvény.

**eventek.c**

1. **void** ProgramEvenetek(SDL\_Renderer \*renderer, SDL\_Window \*window, gomb \*gombok, **int** gombmeret, alakzat \*alakzatok, **int** alakzatmeret, szo \*szavak, **int** szomeret, **bool** loginaktivoldal, **bool** \*programfut, adatoklista \*\*lista, ar \*hotelar);

Ez a program main modul loopján belüli fő loop. Addig loopol ez a függvény, amig oldalváltás nem történik, vagy ki nem lép a felhasználó. Paraméterül kapja a három megjeleníthető objektum tömbjét, és méreteiket, egy bool változót, ami megadja, hogy a login oldal, vagy a felhasználó oldal az aktív, pointerként egy másik bool változót, ami a main modul loopjának feltétele, kilépés esetén false értéket kap, a láncolt listát pointer pointerként, és a hotelárakat. Ez a függvény az egérkattintást, a gomblenyomást, valamint a szövegbevitel eventeket figyeli. Ha triggereli valamelyiket, akkor meghívásra kerül egy másik függvény ami meghatározza, hogy melyik event, és melyik gomb hatására mit kell csinálni az aktív oldallal. Végül az event után lefrissíti a képernyőt.

1. **bool** FajlbeolvasasEsellenorzes(**char** \*autentikacio, **char** \*felhasznalonev, **char** \*jelszo, adatoklista \*\*lista, ar \*hotelar);

Belépésnél kerül meghívásra. Ez a függvény beolvassa a három fájlt, és ellenőrzi, hogy helyes a felhasználónév, és jelszó, ha minden feltétel teljesül, csak akkor ad igazat vissza, s akkor nyílik meg a felhasználói oldal. Az első paraméter azt határozza meg, hogy regisztráció van, vagy belépés. Regisztrációnál létrehozza a fájlokat, belépésnél fájlokból olvas. Paraméterek a felhasználónév, a jelszó, pointer pointerként a láncolt lista, mivel értéket fog kapni, valamint az ár, mivel az is értéket kap.

1. **void** LoginLap(SDL\_Renderer \*renderer, gomb \*gombok, alakzat \*alakzatok, szo \*szavak);

Ez a függvény hívja meg a loginlap inicializálásának függvényeit. Paraméterekben a megjelenítendő objektumok vannak, mivel ezeket továbbadja a többi függvénynek.

1. **void** FelhasznaloLap(SDL\_Renderer \*renderer, gomb \*gombok, alakzat \*alakzatok, szo \*szavak);

Ugyanaz mint a felső függvény, csak a felhasználói oldal értékeivel.

1. **void** Hoteladatok(hotelszobak \*hoteladat);

Feltölti a hotel alapértelmezett adataival a hotelszobak struktúra tömbjét.

1. **bool** Fedes(gomb \*cel, **int** x, **int** y);

Segédfüggvény, meghatározza, hogy a kattintás az adott gombra volt e. Paraméterként kapja a cél gombot, valamint az egér x, y koordinátáját.

1. **int** Gombindex(gomb \*gombok, **int** meret, SDL\_Event \*event);

Végigloopol a gombokon, s a felső függvény segítségével megadja annak a gombnak az indexét amelyre kattintott a felhasználó. Paraméterben kapja a gombok tömböt, a méretét, valamint a regisztrált SDL eventet.

1. **void** LoginOldalGombfunkciok(**int** index, gomb \*gombok, FelhasznaloAdatok \*felhasznalo, SDL\_Event \*event, **int** billentyuevent, **bool** \*lapcsere, adatoklista \*\*lista, ar \*hotelar);

Ez a függvény választja ki, hogy az adott felhasználó eventre, valamint a kiválasztott gombra melyik függvény hívódjon meg a bejelentkezesmenu.c modulból. Ez a függvény akkor lehet aktív ha a login oldal van megnyitva. Paraméterül a kiválasztott gomb indexét, a gombok tömböt, a felhasznaloadatok struktúrát, a regisztrált eventet, egy int számot, ami az event fajtáját, a oldalcsere bool változóját, a láncolt listát, és a hotelárakat kapja meg.

1. **void** FelhasznaloOldalGombfunkciok(**int** index, gomb \*gombok, alakzat \*alakzatok, szo \*szavak, hotelszobak \*szobak, SDL\_Event \*event, ListazandoFoglalasok \*listazandofoglalasok, **int** billentyuevent, adatoklista \*\*lista, ar \*hotelar);

Ugyanúgy mint a login oldalnál, ez a függvény választja ki a megfelelő függvényt a felhasználó oldalnál regisztrált eventekre. Paraméterezése hasonló, itt az alakzat, valamint a szó struktúrát is megkapja a függvény.

**bejelentkezesmenu.c**

1. **void** BelepesRegisztracioGomb(gomb \*gombok, FelhasznaloAdatok \*felhasznalo);

Ez a függvény jeleníti meg kattintásra a felhasználó, és jelszó beviteli mezőt, a vissza és ok gombot. Paraméterül a gombok, valamint egy FelhasznaloAdatok struktúra változót kap, azért, hogy nullázza az eddig beírt adatokat, mivel lehet már volt vissza gomb kattintás.

1. **void** FelhasznaloGomb(gomb \*gombok);

Kitörli a felhasznalonev szót a beviteli mezőből, s engedélyezi a szövegbevitelt.

1. **void** FelhasznaloIras(gomb \*gombok, FelhasznaloAdatok \*felhasznalo, SDL\_Event \*event);

Hozzáfűzi a beviteli mezőhöz, valamint a FelhasznaloAdatok struktúra felhasználónév változójához a beírt betűt, valamint átméretezi a szövegdobozt. Az event paraméterből kapja meg a beírt betűt.

1. **void** FelhasznaloTorles(gomb \*gombok, FelhasznaloAdatok \*felhasznalo, SDL\_Event \*event);

Kitörli a legutolsó beírt betűt a szövegdobozból, és a felhasználónév változóból, valamint újraméretezi a szövegdobozt.

1. **void** JelszoGomb(gomb \*gombok);
2. **void** JelszoIras(gomb \*gombok, FelhasznaloAdatok \*felhasznalo, SDL\_Event \*event);
3. **void** JelszoTorles(gomb \*gombok, FelhasznaloAdatok \*felhasznalo, SDL\_Event \*event);

Ezek a függvények ugyanazt csinálják a jelszóval, mint az előző függvények a felhasználónévvel.

1. **void** VisszaGomb(gomb \*gombok);

Letiltja az aktív gombokat, és újra megjeleníti a bejelentkezes vagy regisztracio gombot.

1. **void** OkGomb(gomb \*gombok, FelhasznaloAdatok \*felhasznalo, **bool** \*lapcsere, adatoklista \*\*lista, ar \*hotelar);

Ez a függvény az ok gomb megnyomására aktiválódik. Paraméterül kapja a gombok tömböt, a FelhasznaloAdatok struktúra változóját, az oldalváltás bool változót, ugyanis ha sikeres az adatok beolvasása, valamint helyes a jelszó és felhasználónév, akkor lapot kell váltani. A láncolt listát pointer pointerként kapja. Beolvasásra a láncol lista, és a hotelár feltöltődik értékekkel.

**felhasznalomenu.c**

1. **void** ListaInicializalas(ListazandoFoglalasok \*listazandofoglalasok);

NULL pointerrel tölti fel a listazandofoglalasok struktúra pointer tömbjét.

1. **bool** Formatum(gomb \*gombok, **int** index);

Ez a függvény a beviteli mezőbe beírt dátumok helyességét és érvényességét ellenőrzi. A megfelelő dátumforma: EEEE.HH.NN Paraméterül a gombok tömböt, és a kijelölt indexet kapja. Igazat ad vissza ha a dátum érvényes és helyes a formátuma.

1. **int** Napszamlalas(**int** \*ev, **int** \*ho, **int** \*nap);

A függvény visszatérési értéke a két dátum között eltelt napok száma. Paraméterei az év, hónap, és nap tömb. A tömb első eleme a -tól dátumot, a második az -ig dátumot kell tartalmazza.

1. **int** Arszamlalas(gomb \*gombok, ar \*hotelar, hotelszobak szoba);

Ez a függvény kiszámolja és visszatér a megadott napokra számított árral. Paraméterként kapja a gombok tömböt, ugyanis innen veszi ki a dátumokat, a hotelárakat tartalmazó struktúra változót, innen tudja, hogy melyik szolgáltatás mennyibe kerül, valamint azt a hotelszobát, aminek számolni kell az árát.

1. **void** Listazas(gomb \*gombok, szo \*szavak, **int** index, **char** \*szoveg);

Ha a gomb nem null pointer akkor az indexel jelölt gomb beviteli mezőjébe írja az adatokat, máskülönben szóként írja ki a megadott indexű helyre.

1. **int** KivalasztottSzoba(gomb \*gombok, alakzat \*alakzatok, **char** \*szobaszam);

Visszatér a kiválasztott szoba indexével. Ezt úgy éri el, hogy végigmegy a szobák gombokon, s ha a kiválasztó négyzet koordinátája megegyezik a szoba gomb koordinátájával akkor azt az indexet adja vissza 0-25 -ig, szóval nem a tényleges gombok tömbben lévő értéket. Ha a szobaszam paraméter nem NULL akkor bemásolja a karaktertömbbe a szobaszámot.

1. **void** AdatTorles(gomb \*gombok, SDL\_Event \*event, **int** index, **int** meret);
2. **void** AdatIras(gomb \*gombok, SDL\_Event \*event, **int** index, **int** meret);

A függvények paraméterként veszi át a gombok tömböt, az SDL eventet, a gomb indexét. Az index elem beviteli mezőjébe lehet írni, illetve törölni.

1. **void** Lezaras(**bool** \*lapcsere);

A függvény az L billentyű hatására aktiválódik, és true értékre állítja a pointerként megadott bool változót, ezért oldalcsere történik.

1. **void** Mentes(adatoklista \*lista);

Az S billentyű hívja meg, elmenti az adatokat a szövegfájlba.

1. **void** FoglalasGomb(gomb \*gombok, alakzat \*alakzatok, szo \*szavak, adatoklista \*\*lista, ListazandoFoglalasok \*listazandofoglalasok);

Tömbökbe és változókba másolja a foglalandó adatokat, majd meghívja a fajladatkezeles.c modulból a foglaló függvényt, végül ha sikeres volt a foglalás megváltoztatja a hotelszobák státuszát. Paraméterül a megjelenítendő objektumokat kapja, mert innen nyeri ki az adatokat, a láncolt listát pointer pointerként, valamint a listazandofoglalasok struktúra változót.

1. **void** TorlesGomb(gomb \*gombok, alakzat \*alakzatok, szo \*szavak, adatoklista \*\*lista, ListazandoFoglalasok \*listazandofoglalasok);

Meghívja a fajladatkezeles.c modul törlő függvényét, és ha sikeres a törlés akkor újralistázza a foglalt szobákat ugyanis, lehet, hogy másik adat bekerül így oda, ahol a törölt adat volt.

1. **void** FoglalasGombMegjelenites(gomb \*gombok);

Ellenőrzi a függvény, hogy minden adat ki van e töltve, valamint a foglalási dátum helyes, s engedélyezi a foglalási gomb megjelenítését.

1. **void** UjFoglalasAdatMegjelenites(gomb \*gombok, szo \*szavak, ar \*hotelar, hotelszobak \*szobak, **int** index);

Szabad szobák kiválasztásánál beírja az adatokhoz a fix értékeket, vagyis a -tól -ig dátumot, és a szoba árát. Paraméterben kapja a gombok tömböt, ugyanis onnan kapja meg a -tól -ig dátumot, a szavak tömböt, ugyanis az egyik szó elembe írja az adatokat, a hotelárakat, a szobákat, valamint a kiválasztott szoba gomb indexét.

1. **void** FoglalasiAdatGomb();

Engedélyezi a szöveg bevitelt.

1. **void** FoglalasAdatMegjelenites(gomb \*gombok, szo \*szavak, **int** index, ListazandoFoglalasok \*listazandofoglalasok, adatoklista \*lista);

Ha foglalt szobát választottunk ki, kilistázza a vendég adatait. Paraméterként kapja a gombok tömböt, ugyanis az adatok egy részét a beviteli mezők egy részébe írja, a szavak tömböt, ugyanis a másik részt, ami fix, ilyen a -tól -ig dátum, valamint az ár, szóként írja ki.

1. **void** FoglaltSzobak(gomb \*gombok, **int** \*datumertek, ListazandoFoglalasok \*listazandofoglalasok, adatoklista \*lista);

Ha helyes -tól -ig dátumot adtunk meg, ez a függvény hívódik meg, ez több másik függvény összessége. Ez a függvény listáz, valamint dönti el, hogy melyik szoba milyen színű legyen.

1. **void** HotelKijelolesVege(alakzat \*alakzatok, gomb \*gombok, szo \*szavak);

Ez a függvény szünteti meg egy szoba kijelölését. Paraméterben kapja az alakzatok tömböt, ugyanis a kijelölő téglalapot el kell tüntetni, a gombok tömböt, mert ha volt a beviteli mezőben valami írva azt is el kell tüntetni, valamint tiltani kell a foglalás vagy törlés gombot, ha meg volt jelenítve. A szavak tömbnél is el kell tüntetni a foglalási adatokat.

1. **void** HotelKijeloles(gomb \*gombok, alakzat \*alakzatok, **int** index);

Odahelyezi a kijelölő téglalapot a kiválasztott gombhoz.

1. **bool** DatumFormatumEllenorzes(gomb \*gombok, **int** \*datumertek);

Összehasonlít két dátumot, és igazat ad vissza ha a az -ig dátum nagyobb mint a -tól dátum. Ezt a függvényt a -tól -ig dátum megadásakor használja a program, s a datumertek pointerben átadja a hívónak a két dátumot, ugyanis az még tovább dolgozik vele.

1. **void** DatumGomb(gomb \*gombok, **int** index);

Kitörli a beviteli mező tartalmát, és elindítja a szövegbevitelt.

1. **void** RadioGombok(**int** index, gomb \*gombok, alakzat \*alakzatok, szo \*szavak, hotelszobak \*szobak);

Ez a függvény akkor hívódik meg, ha beikszel a felhasználó egy szűrőt. A függvény először megnézi, hogy milyen szűrés triggerelte, aztán ha abban a szűréstípusban van másik szűrés, azt deaktiválja, majd összegyűjti, összesen mi alapján vannak a szűrések, végül végigmegy a szoba gombokon, s deaktiválja amelyik nem illik bele a szűrésbe. Paraméterként megkapja a gombokat, az alakzatokat, ugyanis ha ki van választva egy szoba, viszont az nincs benne a szűrésben, meg kell szüntetni a kijelölést, végül a hotelszobak tömb, innen tudja a függvény hogy melyik szoba milyen paraméterekkel rendelkezik.

1. **bool** szuro(hotelszobak \*szoba, **int** ferohelyszuro, **int** klimaszuro, **int** teraszuro, **int** szuroszam);

A felső függvény által beérkezett adatok alapján ez a függvény dönti el egy szobáról, hogy megfelel e a feltételeknek, igazat ad vissza ha igen, hamisat ha nem.

A kódrészletek Syntax Highlighter segítségével formázottak  
https://ndzri.blogspot.com/2016/09/syntax-highlight-code-in-word-documents.html