Dokumentáció

A sakk programom kész állapotánál a grafikus megjelenítés és a bábuk léptetése működik, mellett már a sakk sakkmatt és patt helyzeteket is kiszámolja majd az oda vezető lépéseket el is menti.

Működik a megjelenítés és kattintás, de azon belül a megjelenítésnél az alap_hatter függvény és a hatter_rajzol függvény érdekes bugot alkot vagy működést végez. Az alap_hatter egy 8x8-as bool tömb minden indexénél true-ra állít. Majd a hatter rajzol függvény akkor rajzol ki egy adott sárga négyzetet, ha true. Ennek ellenére true-ra állított tömb elemeket nem jelzi ki sárgaként és a false-ra állított tömb elemeket jelzi ki sárgaként. Ezen a funkció működését nem sikerült kijavítanom.

De nem csak ez a kettő függvénnyel és programrésszel gyarapodott a program. Felépítésében eddig a main.c, a fv.c és fv.h, mentes.c és mentes.h illetve a lepes.c és lepes.h fájlokból áll. A main.c-ben fut a fő programrész és itt főleg a grafikus megjelenítés logikája található jelenleg. A fv.c-ben grafikus és különböző általánosan kellendő függvények helyezkednek el, amit a fv.h-val csatolok össze a main.c-hez. A lepes.c fájlban a bábuk lépéseit leíró (és lépések logikáit kiszámoló) függvények vannak és a lepes.h függvénnyel fűződik a main.c-hez. Ezen fájlok mellé hozzá adódott a mentes.c fájl is, amiben a sakk mentés és memória kezelés elemeit tervezem tárolni.

A main.c fájlban először a kep_betolt függvénnyel 6 képet töltök be az img mappából. Ezek után létrehozom a 8x8-as sakk_tablat, ami a bábuk helyzetét írja le. Létrehoz egy bool tömböt is, ami a sárga négyzetek megjelenését fogja szabályozni. Majd mindkét tömböt alaphelyzetre állítunk az alap_hatter és alap_tabla függvényekkel. A többi függvény (és bool változó a program alaphelyzet állításában kellő.)a beállítja a megjelenő ablakot és megjeleníti azt a sdl_init és SDL_RenderPresent függvényekkel. Ez után a grafikus környezet egérrel való manipulálását írja le. Ez a while ciklusban valósul meg. A ciklus elején a program mindig kirajzolja az adott ciklusban a bábuk állását, lépési lehetőségeit (sárga négyzettel). Ez után az interakciót írja le, ahol kattintás vezérli az eseményeket. Egy kattintásra megnézi, hogy milyen bábúra (vagy a menüben lévő gombra) kattintott a felhasználó és az adott bábu lépési lehetőségeit beírja a sakk_hatter tömbbe (feher_babu_jolepes és stb. függvényekkel). Aztán az első kattintás után a kiválasztva bool igazra állítódik és beleép egy másik ciklusrészre, ahol először megnézi, hogy ugyan arra a bábúra kattintott-e mert ha igen a kiválasztást elveti és alap_hattert állítja vissza, Másodszor megnézi, hogy a kattintott mező az lépő mező és ha az akkor oda lép (illetve a két különleges lépést is itt ellenörzi, ami az elpassant, rosálás és a babu cseréje) és menti azt egy láncolt listába. A többi programrész az ablak bezárását, menü irányítását és kattintás végét írja le.

A fv.c fájlban az első függvény az alap_hatter, ami paraméterként kap egy bool 8x8-as tömböt, majd annak minden elemét true-ra állítja. A második függvény az alap_tabla, ami paraméterként kap egy Babu típusú 8x8-as tömböt (A Babu típus enum-al van leírva a fv.h -fájlban) és azt a sakk szabály szerinti alaphelyzetére állítja. Majd a harmadik a babu_rajzol függvény, ami a paraméterként kap egy renderer pointert (ami a megjelenítő), egy hattér pointert (ami egy kép), egy sakk háttért, ami leírja ,hogy melyik négyzeten igaz és végül az x és y, amik leírják az adott helyet a sakk_hatter két dimenziós tömbben. Ezek után az x és y helyzetű négyzetet kirajzolják a képernyőre, ha igaz az adott helyen a sakk_hatter tömb. A negyedik függvény a kep_betölt függvény, ami paraméterként megkap egy megjelenítőt(SDL_Renderer-t) és egy kép helyét. A kapott információból betölti a kép helyén lévő helyeket. Az ötödik függvény a tabla_rajzol, ahol paraméterként megkapja a SDL_Renderer-t és a táblát (kép) és az ablakba kirajzolja azt. A hatodik függvény az sdl_init, ami szeles és magas int alapján egy széles szer magas ablakot hoz létre arra a pontra ahova a pwindow ponter mutat, illetve létrehoz egy megjelenítőt, ahova a prenderer pointer mutat. A hatodik függvény a babu_kivalaszt, ami a babu_rajzolhoz hasonlóan működik azzal a különbséggel, hogy itt királyok kovételével az osszes fehér vagy fekete babut megjeleniti a menu panelben. A hetedik függvény a menu_rajzol, ahol a menüt rajzolja ki az SDL

könyvtár segítségével.

A lepes.c fájlban, függvények nagy részének célja hogy az adott bábutípus lépését leírja, ennek elérésére kap paraméterben egy Babu típusú sakk_tabla tömböt, ami a bábuk elhelyezkedését adja meg, e-mellett kap egy sakk_lep tömböt, amin kiírja a lehetséges lépéseket és végül megkapja a vizsgált bábu elhelyezkedését (néha a láncolt lista pointerét is megkapják ha kellenek az előző lépések). Majd vannak föggvények amik ezen szabájokra építenek és segítségükkel leírják például a jó lépéseket az adott babuval vagy kiszámolják a sakkot vagy mattot. Az egyik példa erre a feher_babu_jolepes, amiben a jolepes[8][8] paraméter módosítja hogy a jó lehetséges lépéseket írja ki. Ezt úgy teszi, hogy az összes lehetséges lépést lelépi (az adott babuval) és azokat jelöli meg amiket úgy megteheti hogy a király nem lesz sakkban. Azt hogy sakkban van-e azt a sakk fv,-el tudja meg ahol az összes ellenséges ütés mezőt legenerálja és megnézi hogy a király beletartozik-e. A játék végét két matt fügvénnyel tudjuk meg, ahol az összes azonos színű babun alkalmazzuk a feher_babu_jolepes fv.-t és ha egyiknél se igaz, akkor igaz lesz. Ha az ellenség nem tud lépni de nincs sakkban akkor patt helyzet alakul ki és döntetlen lesz. A játék végét a konzol jelzi “Fekete_matt” vagy “Feher_matt” vagy “patt” sor kiírásával.

A mentes.c fájlban a láncolt lista alapfunkciójit írja le. Itt található fölösleges nem használt függvény is, de mivel a programot folytatni tervezem megtartom őket hátha kellenek. A láncolt lista két irányú mert az előre hátra lépetetést ezzel a legegyszerűbb megvalósítani. Ezek mellet fájlkezeléses fv-ek is szerepelnek.
A kettő fő a Mentes és Beolvas fv.-ek. A mentes fv-ben a lista elejére megy, majd onnét minden változást a sakk_tabla tömben észlel (vagy azt hogy mit ütött vagy azt hogy rosalt-e) és kiírja a mentett.pgn fajlba.
A Beolvas fv. nem sikerült a specifikáció-hoz mérten teljesíteni idő hiányában. A fv. csak soronként beolvassa a játék lépéseit és kiírja azt a konzolban, nem tölti be azokat a láncolt listába és ez által nem lehet léptetni sem.

A program sorai firtoktatása helyett, most inkább a felhasználói használatról írnék. A sakk tablan egy kattintással lehet kiválasztani a babut, ami után az összes lehetséges lépést sárga négyzettel jelöli. A masodik kattintással a lépést megvalósítja, ha a sarga negyzetre kattintunk, különben pedik a kiválasztást elveti és újra kezdi a folyamatot. Egy körben először a fehér játékos kezd és csak a fehér bábúk választhatók ki, majd a feketék a fehér lépése után. Ha egy gyalogos a pálya végére ér akkor beváltható egy másik babura a menu panel tetjén az adott babura kattintva. A játék végét a konzol fogja mutatni Feher_matt vagy Fekete_matt vagy Patt kiírásával. A menüben a jobb alsó nyíl gombokkal előre hátra tudjuk léptetni a játékot az eddigi lépések által, de ha a tablara kattintunk, akkor vissza tér a jelenlegi állapotra. Ha a plusz jelre kattintunk, akkor új játékot kezd és elveszik az eddig játék. Ha a mentés gombra kattintunk egyszer akkor kiírja a mentes mappában lévő mentett.pgn fájlba. Ha a betöltés gomra kattintunk a beolvas mappában lévő beolvasando.pgn fajt írja ki a konzolba.

(Ezzel a módszerrel segmentation fault-ot ír a program, ha a királlyal kell lépni. Viszont debugolás módszerével működik, de ez lehet, hogy csak az én rendszeremen probléma.)
A program lefordítása:

```
gcc main.c fv.c lepes.c mentes.c -o out `sdl2-config --cflags --libs` -lSDL2_gfx -lSDL2_ttf -lSDL2_image -lSDL2_mixer
```

A program futtatása (linuxban):

```
./out
```

A program debugolása:

gcc -g main.c fv.c lepes.c mentes.c -o out `sdl2-config --cflags --libs` -lSDL2_gfx -lSDL2_ttf - lSDL2_image -lSDL2_mixer

A program futtatása (linuxban):

gdb out

(Majd a programban layout src és run parancsal lehet debugolni.)