Sakk dokumentáció

**Adatszerkezet**

A program futtatásakor a menu függvény hívódik meg. Innen indul a program. Itt dönti el a felhasználó, hogy mire is szeretné a programot használni. Számokkal tud navigálni a különböző opciók között.

**Program vezérlése, függvények**

A program több kisebb modulra van bontva, az összetartozó részeket külön kódokba vannak szervezve. Ezek a kisebb szerepeket betöltő kódrészek például a lépés, egy elmentett játszma beolvasása és mentése, egy lépése ellenőrzése.

Konstans értékek

Mezo tabla[8][8]

Ez a Mezo mátrix reprezentálja a sakktáblát.

Mezo \*feher\_kiraly, \*fekete\_kiraly

A két király pozíciójához a könnyebb hivatkozás miatt jött létre két Mezo konstans.

Lepes \*lepes

Az eddigi lépéseket tárolja ez a Lepes lista, kezdetben a lepes értéke NULL.

char oszlopok[]

A sakktábla oszlopait tárolja a karaktertömb.

char babubetuk[]

A sakkbábuk betűit tárolja a karaktertömb.

char \*babuk[]

a sakkbábukat tárolja el a "karaktertömb".

Konvertálások

int helyesxy(int\* x, int\* y)

Ellenőrzi, hogy érvényes sorokat adott-e meg a felhasználó (sakk játszma közben).

int betubol\_szamra\_konvertal(char\* betu)

A játékos által megadott betűt egy y koordinátává változtatja.

char \*baburakonvertal(char betu, char szin)

Egy bábunak a betűjéből és a színéből előállítja a bábut.

int tablan\_belul\_van\_e(int x, int y)

Ellenőrzi, hogy a koordináták a táblán belülre mutatnak-e.

int karakterbol\_szamra\_konvertal(char c)

Egy bejövő szám karaktert (char-t) konvertál számra (int) típusúra

Bábuk lépéseit ellenőrző, validáló metódusok

int lo\_lepes(Mezo\* jelenlegi, Mezo\* hova)

Ellenőrzi, hogy a ló léphet-e a *jelenlegi* mezőről a *hova* mezőre. Megvizsgálja az összes lehetséges lépési lehetőséget a *jelenlegi* pozícióról, ha az egyik érték megegyezik a *hova* mezővel 1-et ad vissza a metódus.

int kiraly\_lepes(Mezo\* jelenlegi, Mezo\* hova)

Ellenőrzi, hogy a király léphet e a *jelenlegi* mezőről a *hova* mezőre.

int jobbra\_fel\_lepes(Mezo\* jelenlegi, Mezo\* hova),

int jobbra\_le\_lepes(Mezo\* jelenlegi, Mezo\* hova),

int balra\_fel\_lepes(Mezo\* jelenlegi, Mezo\* hova),

int balra\_le\_lepes(Mezo\* jelenlegi, Mezo\* hova),

int atlosan\_jo\_e(Mezo\* honnan, Mezo\* hova)

Ellenőrzi, hogy átlósan léphet e a *jelenlegi* mezőről a *hova* mezőre egy bábu.

int egyenes\_fel\_lepes(Mezo \*jelenlegi, Mezo \*hova),

int egyenes\_le\_lepes(Mezo \*jelenlegi, Mezo \*hova),

int egyenes\_jobbra\_lepes(Mezo \*jelenlegi, Mezo \*hova),

int egyenes\_balra\_lepes(Mezo \*jelenlegi, Mezo \*hova),

int egyenesen\_jo\_e(Mezo\* honnan, Mezo\* hova)

Ellenőrzi, hogy vízszintesen vagy függőlegesen léphet e a *jelenlegi* mezőről a *hova* mezőre egy bábu.

int jobbra\_fel\_ellenoriz(Mezo\* jelenlegi, char szin),

int jobbra\_le\_ellenoriz(Mezo\* jelenlegi, char szin),

int balra\_fel\_ellenoriz(Mezo\* jelenlegi, char szin),

int balra\_le\_ellenoriz(Mezo\* jelenlegi, char szin),

int egyenesen\_fel\_ellenoriz(Mezo\* jelenlegi, char szin),

int egyenesen\_le\_ellenoriz(Mezo\* jelenlegi, char szin),

int egyenesen\_jobbra\_ellenoriz(Mezo\* jelenlegi, char szin),

int egyenesen\_balra\_ellenoriz(Mezo\* jelenlegi, char szin),

int lo\_van\_e(Mezo\* kiraly),

int paraszt\_van\_e(Mezo\* kiraly),

Segédfüggvények, a sakk vizsgálatában segítenek, ha az egyik érték igaz, az egyik játékos sakkban van.

int sakk\_ellenoriz(Mezo\* feher, Mezo\* fekete)

Megvizsgálja, hogy sakkban van-e az egyik játékos.

int oda\_tud\_e\_lepni\_seged(Mezo\* m, char szin)

Megvizsgálja, hogy ha az *m* mezőn állna a *szin* színű király, sakk lenne-e.

Lépést ellenőrző függvények

int lepes\_ellenorzes(Mezo\* honnan, Mezo\* hova, char szin)

Ellenőrzi a *honnan* mezőről lépésének helyességét a *hova* mezőre.

void paraszt\_cserelese(Mezo\* m)

A paraszt bábu cserélését valósítja meg, amikor beér az ellenfél az ellenfél térfeléhez.

int kiraly\_tud\_e\_lepni(Mezo\* m)

Megvizsgálja, hogy a király tud-e olyan helyre lépni, ahol nincs sakkban.

int pozicio\_cserel(Mezo\* honnan, Mezo\* hova, char szin)

Két mezőt megcserél, ellenőrzi, hogy lehetséges-e ez a csere.

Játék közbeni lépések

void elozo\_lepesek\_kiir(Lepes\* l, int szamol)

Kiírja az utolsó 5 lépést, ha kevesebb lépés van, akkor csak amennyi lépés van eddig, annyit ír ki.

int volt\_e\_paraszt\_csere()

Megvizsgálja a függvény, hogy volt e paraszt bábu cserélése a játszma során.

void visszalepes()

Ez a metódus az egyel előtti lépésre állítja vissza a *sakktáblát*.

Egy játszma

void uj\_jatek(int\* navigal)

Betölti a sakktáblát, innen kezdődik a játék.

void aktualis\_megjelenit(Mezo\* m)

Megjeleníti az aktuális táblát, navigációkat, az utolsó 5 lépést és a soron lévő játékost.

int lepesek\_megszamolasa(Lepes\* l, int ossz)

Megszámolja, hogy a hanyadik lépésnél tart a meccs.

int egy\_lepes(char\* betolt\_e, int betolt\_vege)

Egy lépést valósít meg a metódus, ellenőrzi a lépés helyességét.

void tabla\_betolt()

A sakktáblát kezdő állapotba állítja.

Mentés és beolvasás

void jatek\_mentese()

Elmenti egy szöveges fájlba a játszma lépéseit.

void fajlbair(FILE\* mentes, Lepes\* lepes)

Rekurzívan a *mentes*-be írja a lépéseket.

void jatek\_betolt(int\* navigal)

Betölt egy elmentett játszmát, beolvassa a lépéseit az elmentett játszmának.

Használati útmutató

void hasznalati\_utmutato(int\* navigal)

Egy tájékoztatót jelenít meg a felhasználóknak a játékról.

Hibaüzenet függvény

void hibauzenet(char \*uzenet)

Egy hibaüzenetet jelenít meg, a paraméterként beérkező üzenettel.

Kilépés metódus

void kilepes()

Kilép a programból.

Menü

void menu(int\* navigal)

Innen indul a program, megjeleníti a menüt a függvény.