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.