**Programozói dokumentáció**

**Common – minden olyan alap enumot illetve structot tartalmazó script, amiben az általános/közös dolgok vannak**

enum Color — a bábuk színei (a semmilyen, az a NULL bábunak a színe)

struct Position — x és y koordináták, amelyek a bábuk, illetve a pálya négyzeteinek pozícióját jelöli

struct Piece — egy karaktert felépítő tulajdonságokat tartalmazza

type: Unicode karakterrel kirajzolható típus

color: a bábu színe

read: a típus betűs megfelelője (nagybetű=fehér,kisbetű=fekete)

position: a bábu pozíciója a táblán

lastPos: a bábu legutóbbi pozíciója, ahonnan odalépett, ahol most van (kezdésnek ez egyenlő a position-nel)

struct Tile — a pálya egy területe, amelyen van egy bábu (pozícióját ennek egy 8x8-as mátrix x és y indexe adja)

char\*GetColor() — egy átadott color paraméter string formában való változatát adja vissza

bool IsPosEqual() — megnézi, hogy két pozíció x és y koordinátája megegyezik e

void SetPosEqual() — két pozíciót egyenlővé tesz

bool IsInBound() — megnézi, hogy az átadott koordináták a pályán belülre esnek-e (0<x,y<8)

**Board – a táblához tartozó scriptek vannak benne, pl.: beolvasás, mentés kirajzolás**

char\*\*ReadBoardState() — beolvassa a fájlból a jelenlegi állását a táblának és egy 8x8-as mátrixban visszaadja azt, ha a fájl üres, az alaphelyzetet generálja

Position\*ReadLastPosition() — beolvassa fájlból a táblán lévő bábuk legutóbbi pozícióját és egy Position tömbbe elmenti azokat

Piece\*ReadDownedPieces() — beolvassa fájlból a már leütött bábukat és elmenti őket egy Piece tömbbe

void SaveBoard() — lementi a tábla állását a négyzeteket tartalmazó mátrixból és fájlba írja

void SaveLastPosition() — lementi a táblán lévő bábuk legutóbbi pozícióját

void SaveDownedPieces() — lementi a már leütött bábukat

void ClearSavedBoard() — bekér egy fájlnevet, amelynek tartalmát törli

Tile\*\*InitializeBoard() — a bábuk jelenlegi helyzetéből frissíti a táblát és visszaadja azt

void DrawBoard() — kirajzolja a tábla jelenlegi helyzetét, figyelembe véve a kiválasztott bábut és annak lehetsége lépéseit

void FreeTiles() — felszabadítja a lefoglalt mátrixot

**Piece – minden olyan függvényt tartalmaz, am ia bábukhoz köthető (kiválasztás, lehetséges lépések nézése/korrigálása, promóció, stb...)**

Piece\*InitializePieces() — a beolvasott táblaállás alapján létrehozza a bábukat

Piece\*SelectPiece() — egy megadott pozícióval kiválaszt egy bábut, ami azon a területen van, ha ezt nem tudja megtenni (másik színű bábu van ott/nincs ott bábu) NULL-t térít vissza

void SetPiecesEqual() — egyenlővé tesz két bábut

Piece\*MovePiece() — megváltoztatja egy bábu pozícióját és tényleges mozdulás esetén az utolsó pozícióját is, majd visszaadja a bábukat

Piece\*RemovePiece() — töröl egy bábut egy megadott pozícióból, majd visszaadja az újonnan lefoglalt bábukat

Piece\*MoveOrCapture() — az előző két fügvényt használva mozgat/leüt bábukat, attól függően, hogy mit csinál a felhasználó

Position FindKing() — megkeresi az átadott színű játékos királyát és visszaadja annak pozícióját

Piece NullPiece() — generál egy NullPiece-t aminek minden adata NULL/-1

Piece Promotion() — ha egy gyalog bemegy az ellenfél alapvonalára, promótálja azt a bábut

Piece\*GetPieceFromPosition() — visszaad egy bábut amely egy megadott pozícióban van

Piece\*CreatePiece() — létrehoz egy bábut a megadott adatok alapján

**LegalMoves – a bábuk lehetséges lépéseivel és a sakkal foglalkozó függvényeket tartalmazza**

Position\*GetLegalMoves() — egy megadott bábu leehtséges lépéseit adja vissza egy Position tömbben (itt még nem nézi a sakk által befolyásolt lépéseket, amelyeket egyébként nem tehetne meg)

Position\*CorrectLegalMoves() — itt korrigálja a már lekért lehetséges lépéseket, egyéb körülményeket belevéve, mint pl. sakk, majd visszaadja az frissített Position tömbböt

bool LegalMovesContains() — megnézi, hogy a lehetséges lépések között előfordul-e a megadott pozíció

bool IsInCheck() — megnézi, hogy a király éppen sakkban van-e, azáltal, hogy végigmegy az összes ellentétes színű bábu lehetséges lépésein

**Main – a fő játékmenet itt zajlik, illetve a játékosokkal való kommunikáció nagy része**

Position GetInput() — bekér egy pozíciót a játékostól, majd visszaadja azt

bool CheckMate() — megvizsgálja, hogy éppen matton áll e a soron következő játékos

char\*\*ReadPlayBackBoardState() — egy megadott lépésszám utáni állást tud beolvasni és visszaadni egy mátrixban

void PlayBack() — kirajzolja a legutóbb legjátszott/félbehagyott mérkőzés kezdeti állását és a játékos tud benne lépkedni előre/hátra

**main() — itt történik a játék fő menete**

elsőnek a feldobott opciókból választhat a játékos, hogy új menetet szeretne e kezdeni, folytatni az előzőt, vagy visszanézni egyet, illetve kiír néhány alap instrukciót a játékoz

a kiválasztott “játékmód” alapján létrehozza a bábukat, illetve a táblát

a cikluson belül megkeresi a soron következő játékos királyát és megnézi, hogy sakkban van-e, ha igen, akkor megnézi, hogy mattot kapott-e, ilyenkor befejezi a játékot, ha nem, akkor megnézi, hogy patt helyzet alakult-e ki, ilyenkor szintén befejezi a játékot

ezután bekéri az inputot a játékostól és kiválaszt egy bábut, amennyiben fel szeretné adni/döntetlent szeretne/ki szeretne lépni, azt mind itt teheti meg

lekéri a kiválasztott bábu lehetséges lépéseit, majd korrigálja azokat

hogyha olyan bábut választott ki, amivel nem tud lépni, akkor újat kér tőle

ezek után kirajzolja a tábla állását, itt már a kiválasztott bábut, illetve annak lehetséges lépéseit is beleértve

bekéri a felhasználótól a mozgás céljának koordinátáit, amennyiben változtatni szeretne a jelenlegi szelekción, azt is megteheti (bár ez szembe megy a sakk szabályaival, miszerint, ha már megfogtál egy bábut, azzal kell lépni, viszont itt könnyen félreüthet az ember, vagy akár az is lehet, hogy meg szeretné nézni kirajzolva, hogy pontosan hova tud lépni, még mielőtt bábut választana, ezért úgy döntöttem, hogy megadom ezt a lehetőséget)

amennyiben a bábu lehetséges lépései közt megvan a kivánt cél, odalép, ilyenkor attól függően, hogy azon a területen állt-e már másik bábu, vagy leüti azt, vagy szimplán odalép

ezután megvizsgálja, hogy a játékosok nem ismételnek-e lépéseket, ugyanis 3-3 ismétlés után a játék döntetlen lesz

kirajzolja a lépés utáni állapotot, majd lementi a bábukat és a következő játékos jön