```
////////////////////////////////////////////////////////////////////////
// C++ source code of Wrappers.h Version 8.3.1 at greenchess.com //
////////////////////////////////////////////////////////////////////////

// include the stockfish engine header file //
#include "StockfishEngine.h"

// declare extern c for the c++ compiler so c++ functions can be used as c functions //
#ifdef __cplusplus
extern "C" {
#endif

// include boolean definition file //
#include <stdbool.h>

// wrapper to create new game data //
void *DataNew();

// wrapper to copy game data from source to destination //
void DataCopy(void *Destination, void *Source);

// wrapper to write the pgn data to a game file //
void DataWritePGN(void *Data, const char *FileName, bool BlackMoves);

// wrapper to append the pgn data to a game file //
void DataAppendPGN(void *Data, const char *FileName, bool BlackMoves);

// wrapper to initialize the game data //
void DataInitializeAttributes(void *Data);

// wrapper to store the game data header //
void DataStoreGameHeader(void *Data, const char *WhiteName, const char *BlackName,
                         const char *SiteLocation, const char *EventTitle, const char
                         *RoundNumber, const char *EndResult, const char
                         *DateGamePlayed, const char *TimeControl, const char
                         *WhiteElo, const char *BlackElo, const char *WhiteUSCF,
                         const char *BlackUSCF, const char *Annotator, const char *ECO,
                         const char *Opening, const char *Section, const char
                         *BoardNumber, const char *FENString);

// wrapper to store a move to the game data //
void DataStoreMove(void *Data, const char *MadeMove, const char *Annotation);

// wrapper to get the fen string //
char *DataGetFENString(void *Data);

// wrapper to get the white player name //
char *DataWhitePlayer(void *Data);

// wrapper to get the black player name //
char *DataBlackPlayer(void *Data);

// wrapper to get the time control //
char *DataGetTimeControl(void *Data);

// wrapper to get the opening //
char *DataGetOpening(void *Data);

// wrapper to get the section //
char *DataGetSection(void *Data);

// wrapper to get the board number //
char *DataGetBoardNumber(void *Data);

// wrapper to get the white elo //
char *DataGetWhiteElo(void *Data);

// wrapper to get the black elo //
char *DataGetBlackElo(void *Data);

// wrapper to get white uscf //
char *DataGetWhiteUSCF(void *Data);

// wrapper to get black uscf //
char *DataGetBlackUSCF(void *Data);

// wrapper to get the annotator //
char *DataGetAnnotator(void *Data);

// wrapper to get the eco //
char *DataGetECO(void *Data);

// wrapper to get the site //
char *DataGetSite(void *Data);

// wrapper to get the event //
char *DataGetEvent(void *Data);

// wrapper to get the round //
char *DataGetRound(void *Data);

// wrapper to get the result //
char *DataGetResult(void *Data);

// wrapper to get the date //
char *DataGetDate(void *Data);

// wrapper to get the data uci //
char *DataGetGameDataUCI(void *Data);

// wrapper to get the game data annotations //
char *DataGetGameDataAnnotations(void *Data);

// wrapper to get the data list size from the database //
int DatabaseGetDataListSize(void *Database);

// wrapper to get the game data uci //
char *DatabaseGetGameDataUCI(void *Database, int OpeningBookIndex);

// wrapper to the the game descriptions //
char *DatabaseGetGameDescriptions(void *Database);

// wrapper to create a new database //
void *DatabaseNew();

// wrapper to get game data //
void *DatabaseGetGameData(void *Data, void *Database, int GameNumber);

// wrapper to read in a game collection //
void DatabaseReadGameCollection(void *Database, const char *GameFile);

// wrapper to create a new opening book //
void *OpeningBookNew();

// wrapper to initialize the opening book attributes //
void OpeningBookInitializeAttributes(void *OpeningBook);

// wrapper to store opening book data //
void OpeningBookStoreOpeningBookData(void *OpeningBook, unsigned long long
                                     ZobristHash, const char *NextMove);

// wrapper to see if a move exists in the opening book //
bool OpeningBookMoveExists(void *OpeningBook, unsigned long long ZobristHash);

// wrapper to get the next move in the opening book //
char *OpeningBookGetNextMove(void *OpeningBook);

// wrapper to create a new board //
void *BoardNew();

// wrapper to copy board from source to destination //
void BoardCopy(void *Destination, void *Source);

// wrapper to initialize the board attributes //
void BoardInitializeAttributes(void *Board, const char *FENLayout, const char
                               *FENCastlingRights, const char *FENColor, const char
                               *FENEnpassantSquare);

// wrapper to make the given move on the board //
void BoardMakeMove(void *Board, int Depth);

// wrapper to determine if king was captured //
bool BoardKingIsCaptured(void *Board);

// wrapper to find the given move on the board //
bool BoardFindMove(void *Board, const char *Move);

// wrapper to determine if board is white to move next //
bool BoardIsWhite(void *Board);

// wrapper to determine if board is black to move next //
bool BoardIsBlack(void *Board);

// wrapper to determine if the game is over yet //
bool BoardGameOver(void *Board);

// wrapper to determine if there is a checkmate //
bool BoardIsCheckmate(void *Board);

// wrapper to determine if there is a stalemate //
bool BoardIsStalemate(void *Board);

// wrapper to determine if threefold repetition has occurred //
bool BoardIsThreefoldRepetition(void *Board);

// wrapper to determine if there is insufficient material for mate //
bool BoardInsufficientMaterialToMate(void *Board);

// wrapper to determine if an en passant move was just made on the board //
bool BoardIsEnpassant(void *Board);

// wrapper to determine if the board is in check //
bool BoardIsInCheck(void *Board);

// wrapper to get the uci board label for the move //
char *BoardUCILabel(void *Board);

// wrapper to get the pgn board label for the move //
char *BoardPGNLabel(void *Board);

// wrapper to get the list of capture squares //
char *BoardGetListOfCaptureSquares(void *Board);

// wrapper to get the list of threatened squares //
char *BoardGetListOfThreatenedSquares(void *Board);

// wrapper to get the zobrist hash for the current board position //
unsigned long long BoardGetZobristHash(void *Board);

// wrapper to create a new engine //
void *EngineNew();

// wrapper to initialize the transposition table //
void EngineInitializeTransTable(void *Engine);

// wrapper to make the engine stop //
void EngineMakeEngineStop(void *Engine);

// wrapper to compact the transposition table //
void EngineCompactTransTable(void *Engine, bool AgeForward);

// wrapper to find the best move on the board //
char *EngineFindBestMove(void *Engine, void *Board, int TimeLimit, int PlyLimit);

// wrapper to find if there is only one possible move on the board //
bool OnlyOnePossibleMove(void *Board);

// finish extern c for the c++ compiler so c++ functions can be used as c functions //
#ifdef __cplusplus
}
#endif
```