PL_PIG_CHESS_INTERFACE Package Body

Method Signatures:

CREATE OR REPLACE PACKAGE BODY PL_PIG_CHESS_INTERFACE

This is the main package body declaration for the PL_PIG_CHESS_INTERFACE package. It contains various procedures and functions related to interacting with the chess engine and performing tests.

PROCEDURE WR(s VARCHAR2)

This procedure writes a string `s` to the output. It is used for displaying messages and results.

FUNCTION SET_IN(members BINARY_INTEGER, setM BINARY_INTEGER) RETURN BOOLEAN

This function checks if a member is part of a set. It takes two binary integers, `members` and `setM`, as input and returns `TRUE` if `members` is a subset of `setM`, and `FALSE` otherwise.

FUNCTION MOVETXT(trk, piece, english) RETURN varchar2

This function converts an internal move format to a text format. It takes the move data (`trk`), the piece involved (`piece`), and a boolean flag indicating if the output should be in English (`english`) as input. It returns the move in text format.

FUNCTION SHORT(mvtxt, Multiple) RETURN VARCHAR2

This function creates a short version of a move text. It takes the move text (`mvtxt`) and a boolean flag indicating if multiple moves are allowed (`Multiple`) as input. It returns a shortened version of the move text.

PROCEDURE OUTPUT_POSITION(english)

This procedure outputs the current game position to the console. It takes a boolean flag indicating if the output should be in English (`english`) as input.

FUNCTION EPD_STR(operationlist) RETURN varchar2

This function returns the current position in EPD (or FEN) format. It takes an optional operation list (`operationlist`) as input and returns the position in the specified

format.

FUNCTION INV_EPD(epdfenstr) RETURN VARCHAR2

This function returns a reversed EPD (or FEN) position. It takes an EPD or FEN string (`epdfenstr`) as input and returns the reversed position.

FUNCTION POSITION_STR(english) RETURN varchar2

This function converts the internal position array format to the internal string format for a position. It takes a boolean flag indicating if the output should be in English (`english`) as input and returns the position in string format.

PROCEDURE NEW_GAME(White, Black, STARTPOSITION, p_TheoryMode, p_InteractionMode)

This procedure starts a new chess game. It takes the engine strength for White (`White`) and Black (`Black`), the starting position (`STARTPOSITION`), the theory mode (`p_TheoryMode`), and the interaction mode (`p_InteractionMode`) as input. It initializes the game and sets up the necessary variables.

PROCEDURE DO_BOTMOVE(OverruleLevel)

This procedure makes a move for the engine. It takes an optional override level (`OverruleLevel`) as input. It performs the engine's move and updates the game state.

PROCEDURE DO_BOTGAME(maxmoves)

This procedure plays a game between two engine instances. It takes the maximum number of moves (`maxmoves`) as input. It alternates between the two engines making moves until the game ends.

PROCEDURE DO_MOVE(fromto)

This procedure allows a human player to make a move. It takes the move in algebraic notation (`fromto`) as input. It validates the move and updates the game state accordingly.

PROCEDURE SET_White(White)

This procedure sets the engine strength for the White player. It takes the engine strength (`White`) as input and updates the corresponding variable.

PROCEDURE SET_Black(Black)

This procedure sets the engine strength for the Black player. It takes the engine strength (`Black`) as input and updates the corresponding variable.

PROCEDURE TAKEBACK_MOVE

This procedure takes back the last move. It is not yet implemented.

PROCEDURE TAKEBACK_MOVES

This procedure takes back the last two moves. It is not yet implemented.

PROCEDURE test1

This procedure is a test procedure that performs a series of moves and evaluations. It demonstrates the usage of various functions and procedures in the package.

PROCEDURE test2

This procedure is another test procedure that performs pre-processing and displays the positional data. It is used for debugging and testing the engine's evaluation.

PROCEDURE WRMOVES

This procedure writes the engine's move and best/secondary/avoid moves to the output. It is used to display the engine's recommendations.

PROCEDURE CLCPOINTS(p_points)

This procedure calculates points by matching the engine's move with the test suite's best/worst move. It takes the current points (`p_points`) as input and updates it based on the match.

PROCEDURE test_BKtest(lvl, poslow, poshigh)

This procedure runs the B-K test, a set of test positions to evaluate the engine's performance. It takes the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

PROCEDURE test_MSquickTest(lvl, poslow, poshigh)

This procedure runs the Quicktest by Michael Scheidl, a set of test positions to evaluate the engine's performance. It takes the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

PROCEDURE test_THmyPosTest(lvl, poslow, poshigh)

This procedure runs the MY POSITIONAL TEST SUITE by Tony Hedlund, a set of test positions to evaluate the engine's positional understanding. It takes the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

PROCEDURE test_SLendgameTest(lvl, poslow, poshigh)

This procedure runs the Endgame testsuite by Sune Larsson and John Nunn, a set of test positions to evaluate the engine's endgame skills. It takes the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

PROCEDURE test_CCRTest(lvl, poslow, poshigh)

This procedure runs the One Hour Test by Larry Kaufman, a set of test positions to evaluate the engine's performance. It takes the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

PROCEDURE test_ColditzTest(lvl, poslow, poshigh)

This procedure runs the Colditz test suite by Ferdinand Mosca, a set of test positions to evaluate the engine's performance. It takes the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

PROCEDURE test_BBCTest(lvl, poslow, poshigh)

This procedure runs the Big Book Of Combinations, a set of test positions to evaluate the engine's tactical skills. It takes the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

PROCEDURE test_ReinfeldTest(lvl, poslow, poshigh)

This procedure runs Reinfeld's tactical positions, a set of test positions to evaluate the engine's tactical understanding. It takes the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

PROCEDURE test_LCTIITest(lvl, poslow, poshigh)

This procedure runs the LCT II (Louguet Chess Test II) by Frdric Louguet, a set of test positions to evaluate the engine's positional and tactical skills. It takes the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

PROCEDURE test_SBDTest(lvl, poslow, poshigh)

This procedure runs the Silent but Deadly (sbd) test suite, a set of test positions to evaluate the engine's tactical skills. It takes the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

PROCEDURE test_STSTest(suite, lvl, poslow, poshigh)

This procedure runs the Strategic Test Suite (STS), a set of test positions to evaluate the engine's strategic understanding. It takes the suite number (`suite`), the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

PROCEDURE test_PIG(lvl, poslow, poshigh)

This procedure runs the Pig-chess found errors test suite, a set of test positions to evaluate the engine's performance. It takes the engine strength (`lvl`), the starting position index (`poslow`), and the ending position index (`poshigh`) as input.

Code Behavior:

The PL_PIG_CHESS_INTERFACE package body provides an interface for interacting with the chess engine and performing various tests. It allows users to start a new game, make moves, set engine strengths, and run different test suites.

The `WR` procedure is used to write messages and results to the output. The `SET_IN` function checks if a member is part of a set, which is used for move type comparisons. The `MOVETXT` function converts an internal move format to a text format, allowing for human-readable move descriptions.

The `SHORT` function creates a short version of a move text, which is useful for comparing moves with test suites. The `OUTPUT_POSITION` procedure displays the current game position to the console, providing a visual representation of the board.

The `EPD_STR` function returns the current position in EPD (or FEN) format, allowing for easy sharing and comparison of positions. The `INV_EPD` function reverses an EPD or FEN position, which can be useful for testing or analysis.

The `POSITION_STR` function converts the internal position array format to a string format, providing a textual representation of the position. The `NEW_GAME` procedure starts a new chess game, initializing the game state and setting up the engine strengths and interaction mode.

The `DO_BOTMOVE` procedure makes a move for the engine, performing the engine's move and updating the game state. The `DO_BOTGAME` procedure plays a game between two engine instances, alternating between the engines until the game ends.

The `DO_MOVE` procedure allows a human player to make a move, validating the move and updating the game state accordingly. The `SET_White` and `SET_Black` procedures set the engine strength for the White and Black players, respectively.

The `TAKEBACK_MOVE` and `TAKEBACK_MOVES` procedures are not yet implemented, but they are intended to take back the last move or the last two moves, respectively.

The various `test_*` procedures run different test suites, such as the B-K test, Quicktest, MY POSITIONAL TEST SUITE, Endgame testsuite, One Hour Test, Colditz test suite, Big Book Of Combinations, Reinfeld's tactical positions, LCT II, Silent but Deadly (sbd) test suite, Strategic Test Suite (STS), and Pig-chess found errors test suite. These procedures evaluate the engine's performance in different aspects of chess, including positional understanding, tactical skills, endgame skills, and more.