Code Documentation

Chess Game Designed in C Language

By
Mohamed Sayed Mohamed Abdo
Md Ruhul Kuddus Saleh

Content

Libarries:

#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include <stdbool.h>

Constructs:

For loop
While loop
If statement

Data Types Used:

INT
CHAR
BOOL
VOID
INT[ ][ ]
CHAR[ ][ ]

Functions:

void display()
void change(int r1, int c1, int r2, int c2)
void pawn(int r1, int c1)
void pawnb(int r1, int c1)
void rook(int r1, int c1)
void rookb(int r1, int c1)
void horse(int r1, int c1)
void horseb(int r1, int c1)
void camel(int r1, int c1)
void camelb(int r1, int c1)
void king(int r1, int c1)
void kingb(int r1, int c1)
void queen(int r1, int c1)
void queenb(int r1, int c1)
void player1()
void player2()
int check(int, int);
int check2(int, int);
bool is_king_alive(char K, char board[8][8]);
bool is_kingb_alive(char k, char board[8][8]);
bool is_king_in_check(char king, char board[8][8]);
bool is_kingb_in_check(char king, char board[8][8]);
int main(){…}

Libraries Used

#include<stdio.h>

Printf()
Scanf()

#include<stdlib.h>

System()

#include<conio.h>

Getch()

#include <stdbool.h>

Bool()
True()
False()

Construction

For loop

A for loop is a control flow statement that allows you to execute a block of code repeatedly based on a specific condition. It consists of three parts:

- Initialization: Typically initializes a counter variable. This part is executed only once at the beginning of the loop.
- Condition: Defines the condition that must be true for the loop to continue iterating.
- Update: Typically increments or decrements the counter variable. This part is executed after each iteration of the loop.

While loop

A while loop is another control flow statement that repeatedly executes a block of code as long as a specified condition is true. It consists of a single condition:
The loop continues to execute as long as this condition remains true.

If statement

An if statement is a conditional statement that executes a block of code if a specified condition is true. It can optionally include an else block that executes if the condition is false.

Data Types Used

int:
An int data type in C represents signed integers, typically stored using 32 bits (4 bytes) of memory on most platforms. It encompasses whole numbers, both positive and negative, without any fractional or decimal part.

char:
In C, the char data type denotes single-byte characters, adhering to the ASCII or extended ASCII character encoding standards. It is utilized for representing individual characters such as letters, digits, or symbols.

bool:
The bool data type, though not native to C, can be simulated using the stdbool.h header. It embodies logical values, conventionally represented as true or false, with true corresponding to the integer value 1 and false to 0.

void:
In C, void serves as a type specifier indicating the absence of any specific type. It is prominently used in function declarations to denote functions that do not return a value (void functions) or in pointer declarations for pointers that lack a specific data type (void pointers).

Int[ ][ ]:
This refers to a two-dimensional array of integers in C, providing a structured arrangement of integer values across rows and columns. Each element within the array is referenced by two indices, one for the row and another for the column, commonly used to represent data grids like the chessboard in the provided code.

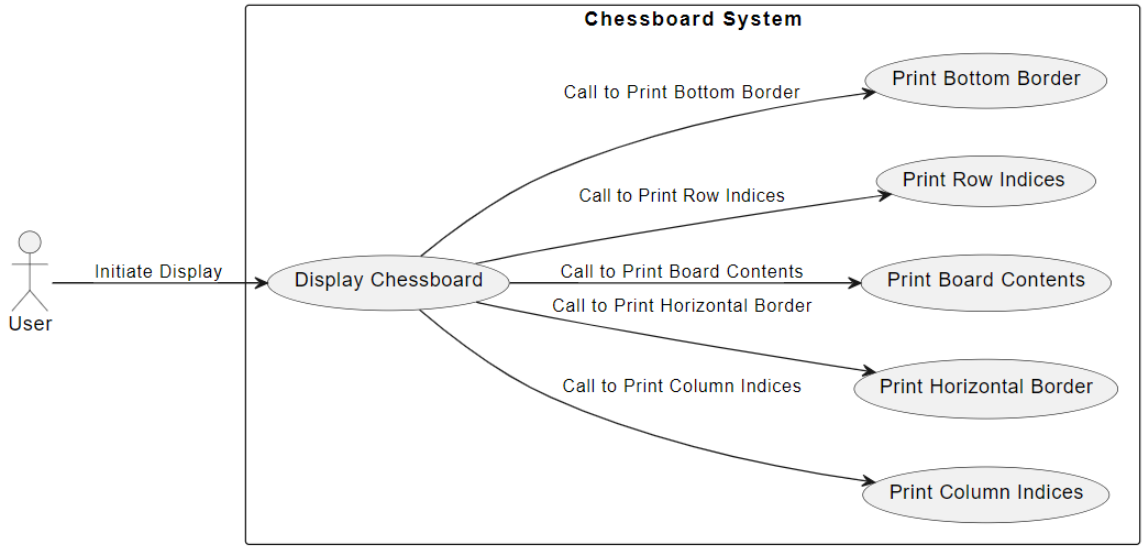
Char[ ][ ]:
Similarly, a two-dimensional array of characters in C organizes character data into rows and columns. Each cell in the array is identified by two indices, typically representing a position on a grid or matrix. In the context of the code snippet, it represents the chessboard, with each character symbolizing a chess piece.

Functions

`void display()`

This function is responsible for displaying the current state of the chessboard to the console. It iterates over the `board` array and prints out the characters representing the pieces and empty squares on the board.

These functions are essential for the gameplay as they handle the visual representation of the chessboard and the updating of the board after each move.

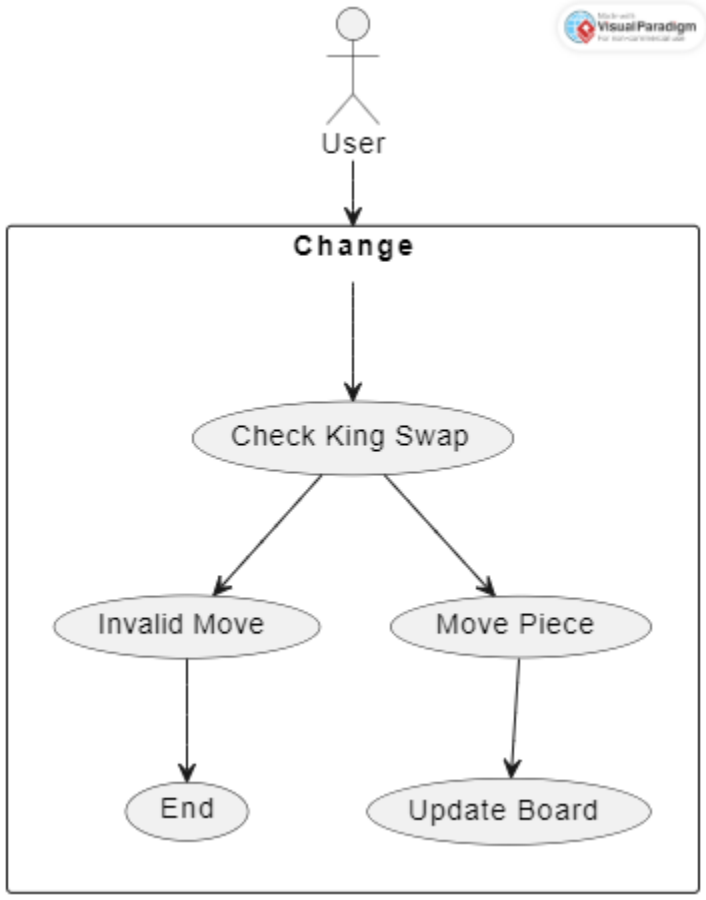


`void change(int r1, int c1, int r2, int c2)`

This function is responsible for updating the chessboard based on the move made by a player. It takes four parameters:

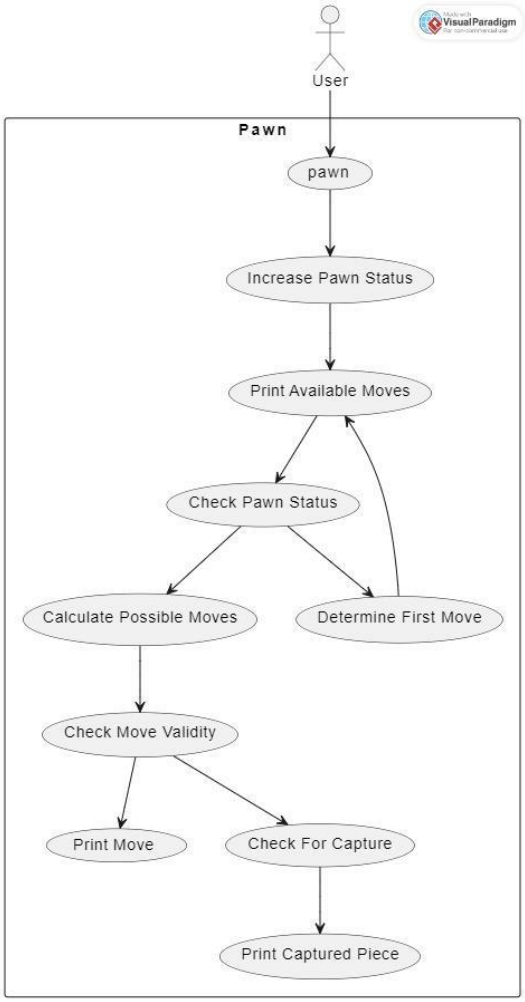
- `r1`: The initial row position of the piece to be moved.
- `c1`: The initial column position of the piece to be moved.
- `r2`: The final row position where the piece will be moved.
- `c2`: The final column position where the piece will be moved.

The function updates the board array to reflect the movement of the chess piece from its initial position to its final position.



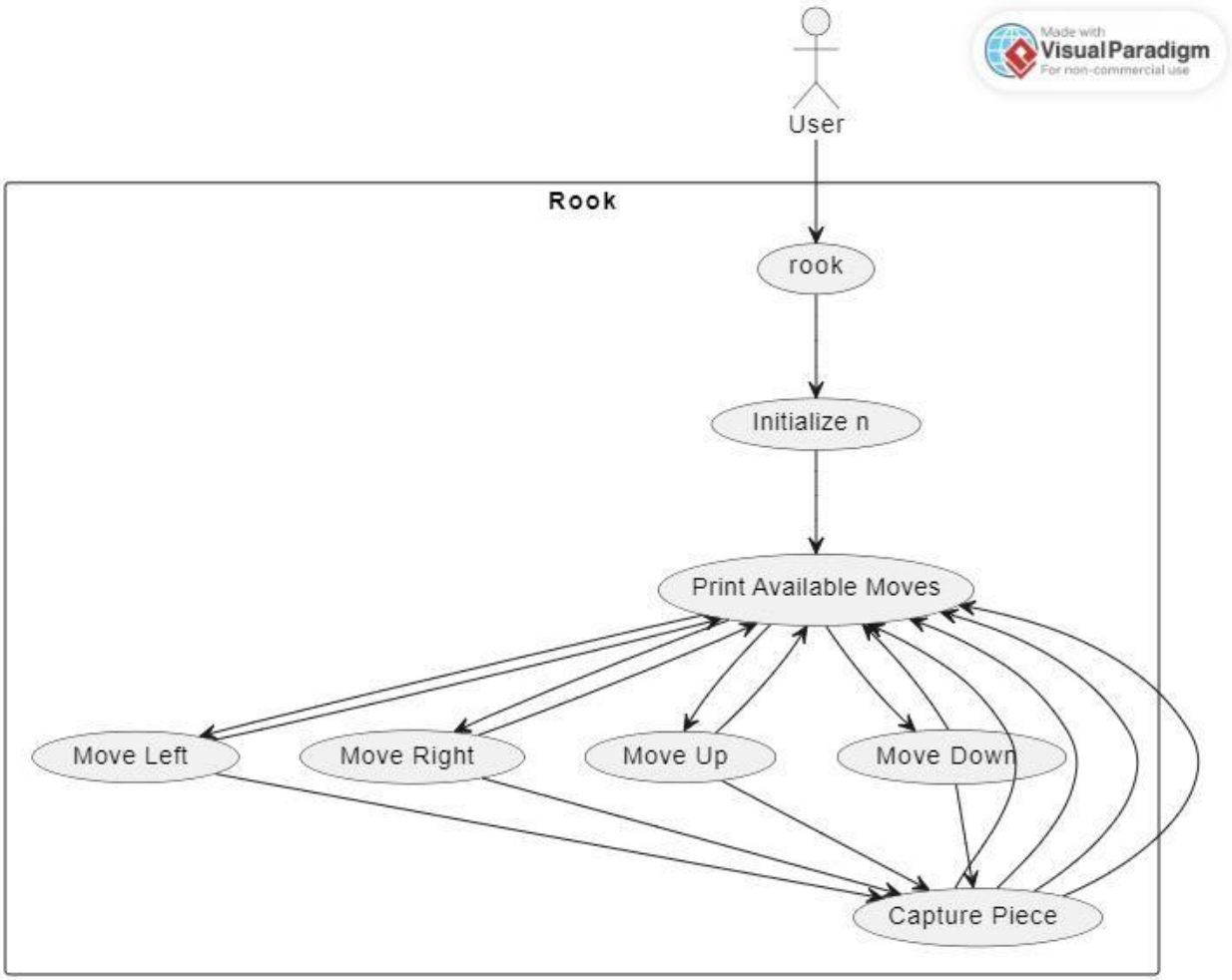
void pawn/pawb(int r1, int c1)

Calculates the available moves for a pawn owned by player 1 (Big Case) and player2 (Small Case) on the chessboard. This function considers the pawn's unique movement rules, including initial double steps and diagonal capturing opportunities.



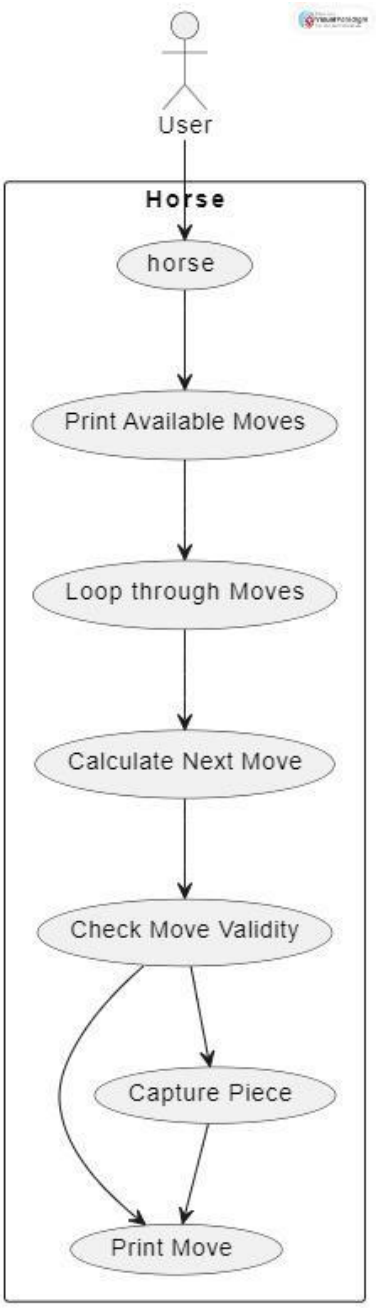
void rook/rookb(int r1, int c1)

Determines the possible moves for a rook (castle) under the control of player 1 (Big Case) and player2 (Small Case) on the chessboard. It evaluates potential movements along horizontal and vertical lines, identifying opportunities for capturing opponent pieces.



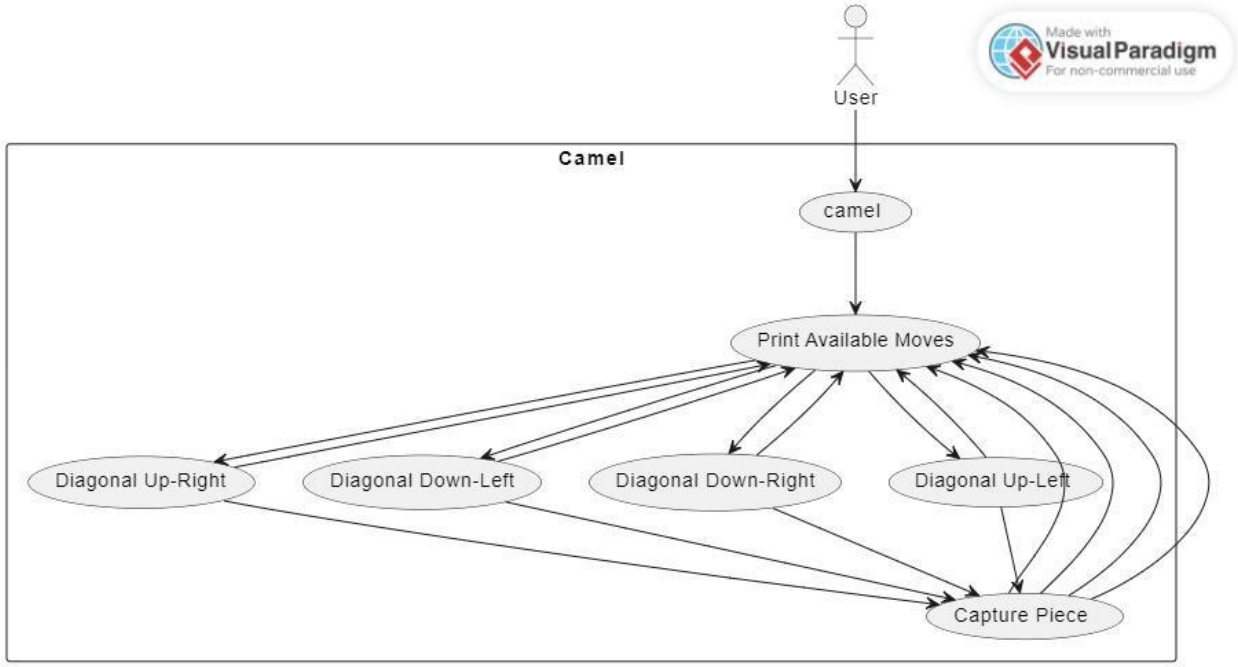
void horse/horseb(int r1, int c1)

Analyzes the available moves for a knight belonging to player 1 (Big Case) and player2 (Small Case) on the chessboard. Knights exhibit a distinct L-shaped movement pattern, enabling them to bypass obstacles and capture enemy pieces effectively.



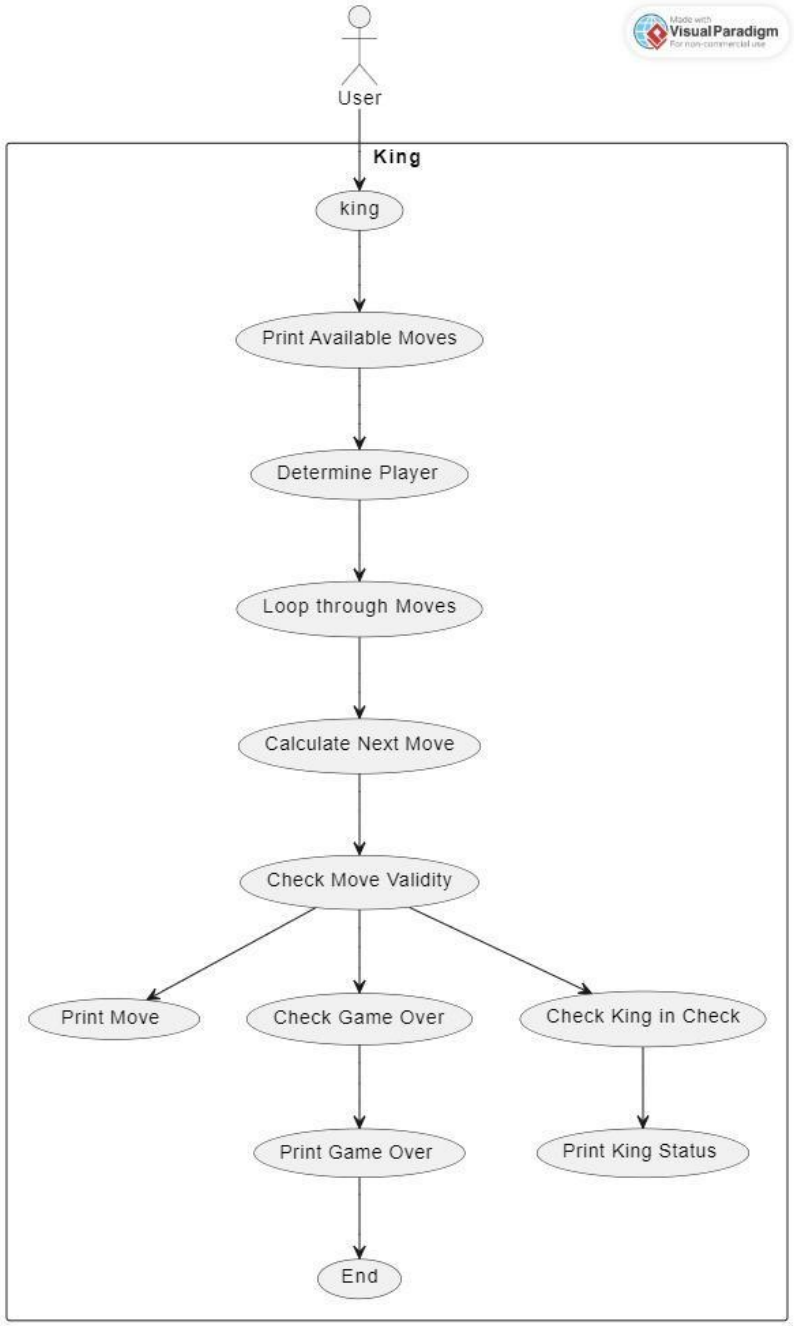
`void camel/camelb(int r1, int c1)`

Calculates the available moves for a bishop (commonly known as a camel) controlled by player 1 (Big Case) and player2 (Small Case) on the chessboard. It evaluates diagonal paths to identify vacant squares and potential captures of opponent pieces.



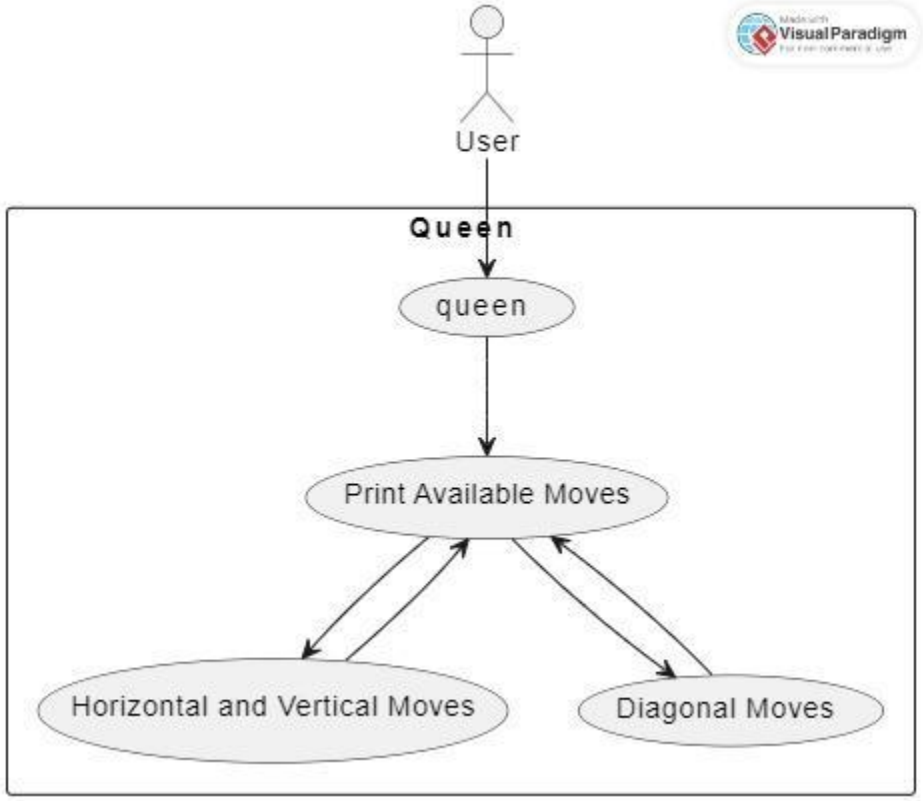
void king/kingb(int r1, int c1)

Identifies the available moves for the king of player 1 (Big Case) and player2 (Small Case) at a specified position on the chessboard. This function considers the king's ability to move one square in any direction while ensuring it remains out of check.



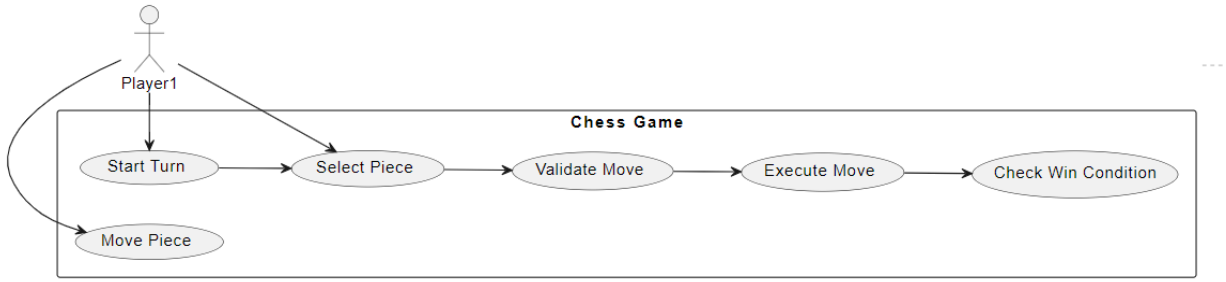
void queen/queenb(int r1, int c1)

Determines the possible moves for the queen controlled by player 1 (Big Case) and player2 (Small Case) on the chessboard. Combining the movement capabilities of rooks and bishops, the queen assesses horizontal, vertical, and diagonal paths for movement and capturing.



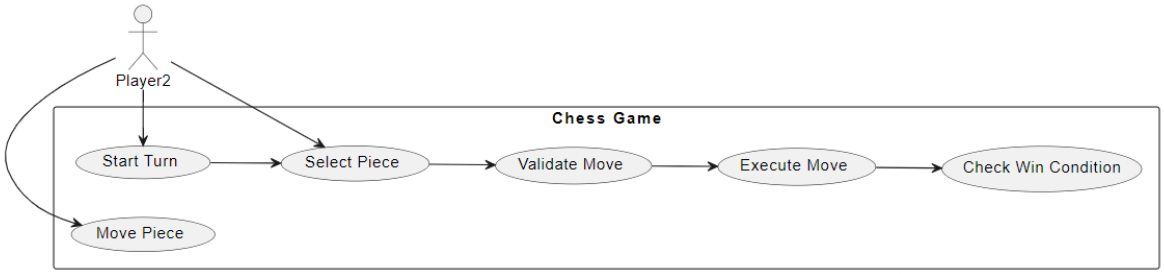
`void player1()`

Manages the actions of player 1 (Big Case) during their turn in the game. This function prompts player 1 to select a piece to move and a destination square, then updates the game state accordingly.



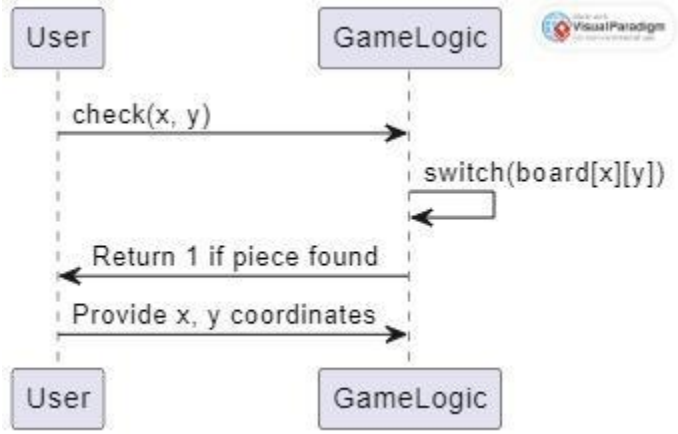
void player2()

Controls the actions of player 2 (Small Case) during their turn in the game. Similar to player 1, it prompts player 2 to choose a piece and a destination square, then updates the game state based on their input.



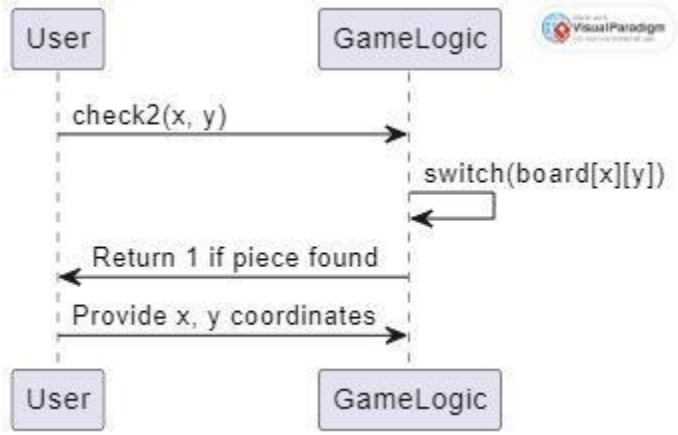
int check(int x, int y)

Checks whether a specific square on the chessboard contains a piece belonging to player 1 (Big Case). This function returns 1 if a piece is found, indicating the presence of player 1's piece, and 0 if the square is empty or contains a piece belonging to player 2.



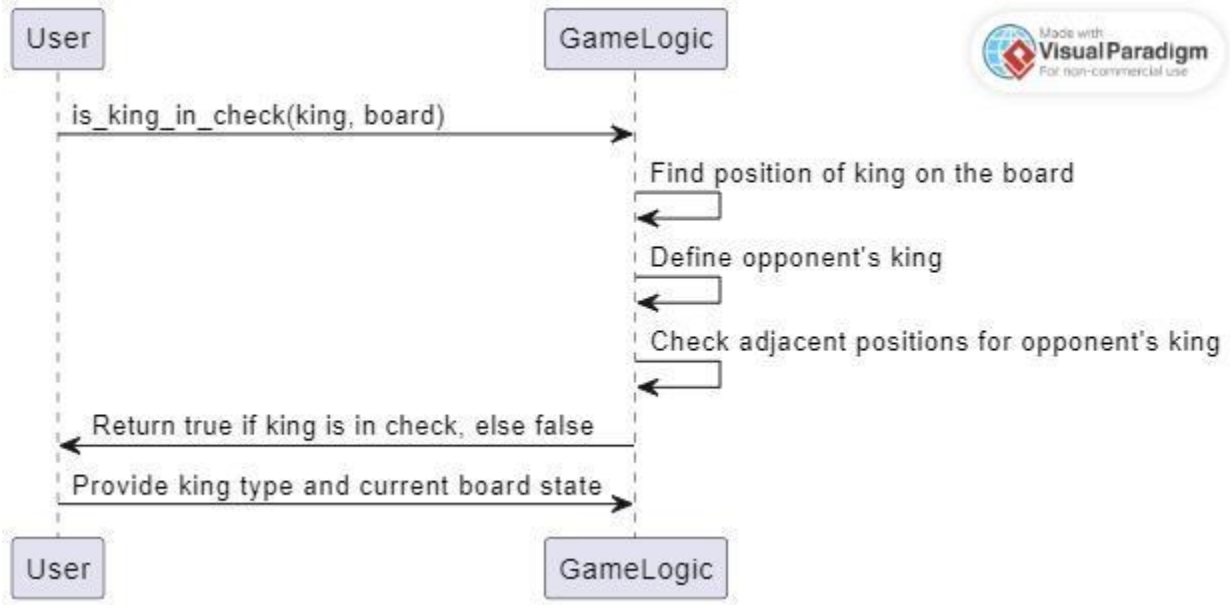
int check2(int x, int y)

Determines whether a particular square on the chessboard contains a piece owned by player 2 (Small Case). Similar to check(), this function returns 1 if a piece is present, indicating player 2's ownership, and 0 if the square is vacant or occupied by a piece belonging to player 1.



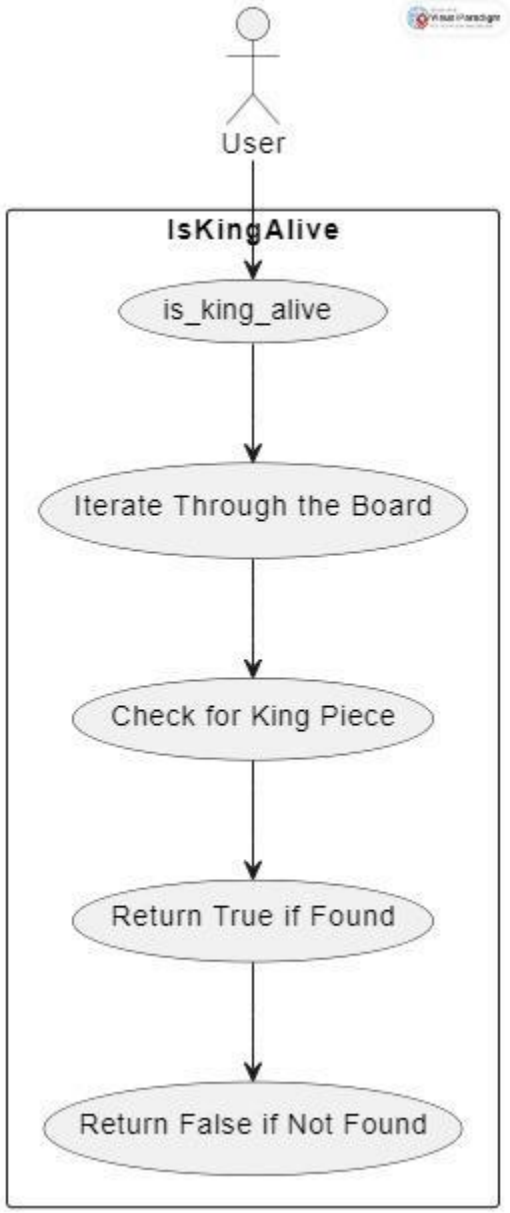
bool is_king/kingb_in_check(char king, char board[8][8])

Evaluates whether the king of player 1 (Big Case) and player2 (Small Case) is currently in a checked position on the chessboard. This function examines the surrounding squares to determine if any opponent pieces threaten the king's safety, returning true if the king is in check and false otherwise.



`bool is_king/kingb_alive(char K, char board[8][8])`

Verifies whether the king specified by the parameter 'K' is still present on the chessboard. This function searches for the specified king piece within the board array, returning true if the king is found (alive) and false if it's not present (eliminated).



`int main(){...}`

The int main(){...} function in the code serves as the central control unit for the program.Main function responsible for initializing the game and managing its flow. The function starts by prompting the user to choose between playing as Player 1 (big case) or Player 2 (small case). It then enters a loop where each player takes turns making moves until the game ends. Inside the loop, each player is prompted to input the position of the piece they want to move and the destination position. The function calls corresponding player functions based on the chosen piece and its movement rules. After each move, the function checks if either player's king has been captured. If so, it declares the opposing player as the winner. The function also includes auxiliary functions for checking if a king is in check and determining if the game has ended. Once the game loop ends, the function returns 0 to indicate successful execution.