**Report:** Data Structure’s FISAC 1 Assignment

**Name:** Deepesh Adwani

**Registration Number:** 210968198

**Section:** B

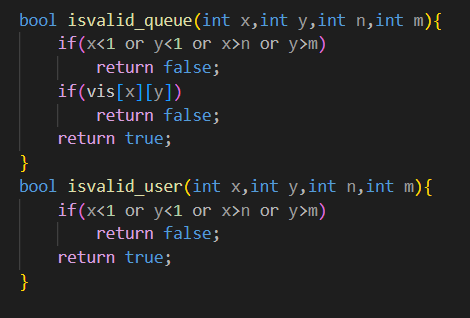
**Knight’s Travel**

**Aim:** The aim of this project is to find the minimum number of moves required to move the knight to the desired position in the game of chess using the shortest path.

**Objectives:** Data Structures used are Queues, Arrays, and Tress.

**Methodology:**

Validation of each step is very necessary for the successful and smooth functioning of the project.

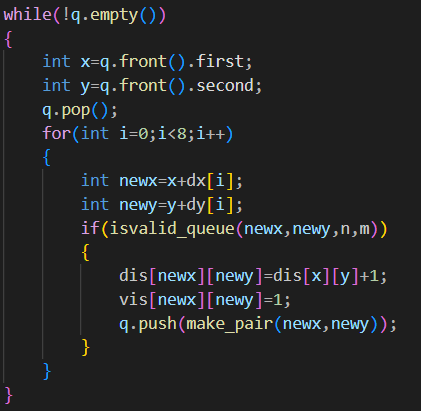


Here, by using the isvalid\_queue function, firstly we are checking that the coordinates entered by the user are in the range of the chess board or not, as we are starting from (1,1) and the end coordinates of the board are (n,m) so the coordinates should lie is this range else it will return false. The vis function checks whether we have visited the entered coordinates previously or not, if visited previously then it returns false. This valid function is used for the queue to be made and for the elements entered in the queue. The function isvalid\_queue will explicitly validate the coordinates used in the queue.

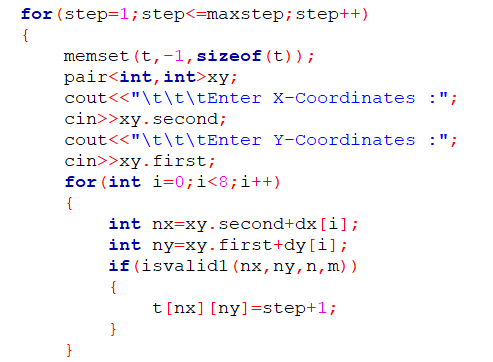
The usage of the isvalid\_user function is only for the input given by the user, it is used to validate the user input. Here also we are checking whether the entered coordinates are in the range of the board or not, because the condition is that knight cant move outside the confinements of the chessboard. If both functions satisfy the conditions, then the program moves ahead by returning true. So, here we are using a Boolean variable in both snippets to validate the coordinates.

But here when we use these two functions then these two do not validate the coordinates entered by the user, let’s say the current position of the knight is (2,2) so the possible moves to reach the final point should be (4,1), (4,3), (1,4), (3,4) now if the user enters any of these coordinates or any other set of coordinate then also the isvalid functions will still give true, even if the input given by the user is not possible. To verify whether the user input is a valid move or not, we use other functions in the main for loop.

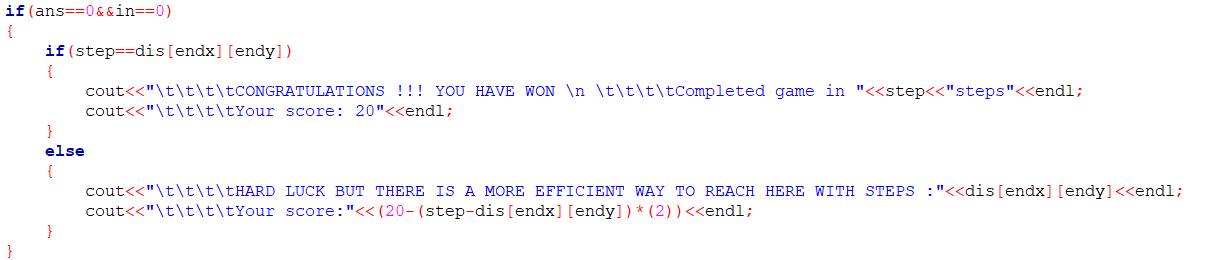
One of the main reasons to make a validation function especially for the queue is that here in this program we are calculating the shortest distance for which minimum number of moves should be done, so to achieve that no location or coordinate should be accessed twice if it occurs in the shortest path. That’s why while at the time when we add elements we set it’s vis[][] coordinate to be equal to 1 which signifies that the coordinate has been visited once.



Now, in this while loop, we will replace the old x and y coordinates with new coordinates i.e., newx and newy. Then we will pass these coordinates and the maximum ranges (n,m) to the isvalid\_queue function and will again check for its validness.



In this for loop we are again using the invalid\_user function because now the user does not have to follow the shortest path, and can move back and forth as many times unless he exhausts the maximum number of moves allowed.



In this step, we are checking whether the number of moves that user played is equal to the minimum moves to reach the final point or not, if it is equal to the minimum number of moves then it displays that you won the game else it displays that there is a more efficient way to reach here with less number of steps.

**Conclusions:** From this project, we got to learn to check validness of every move in the game of chess, we used the arrays and queues data structures also in our program and learned how these data structures work.

**References:** CppReference