
***School of Mechanical and Manufacturing Engineering (SMME),***

***National University of Engineering and Technology (NUST),***

***Sector H-12, Islamabad.***

**Program:** BE-Aerospace **Section:** AE-01

**Session:** Fall 2023 **Semester:** 1st

**Course Title:** Fundamentals Of Programming **Course Code:** CS-109

# ***FOP LAB PROJECT (TIC TAC TOE)***

## ***GROUP MEMBERS:***

- ***M.Mushaf Khan (459299)***
- ***Al Ameerah Bint e Waqar (480966)***
- ***Rumman Tanvir (482064)***

## ***ABSTRACT***

The C++ code written implements a simple console-based Tic-Tac-Toe game for two players. The game utilizes a 1D array to represent a 3x3 game board and features functions for board initialization, display, player turns, and game outcome checking (win or draw). The main game loop prompts players to input positions to place their **"X"** or **"O"** marks on the board, and the game continues until a player wins or the game ends in a draw. The code also includes a visual representation of the board positions and welcomes players to the game.

## ***CODE BREAKDOWN***

The code contains four user-defined functions and a main function. The four user-defined functions are:

1. **void** initializeBoard() → Initializes the board with empty spaces.
2. **void** displayBoard() → Displays the current status of the board.
3. **bool** gameOver() → Checks if the game is over (win or draw).
4. **void** playGame() → Main game loop.

### ***main FUNCTION***

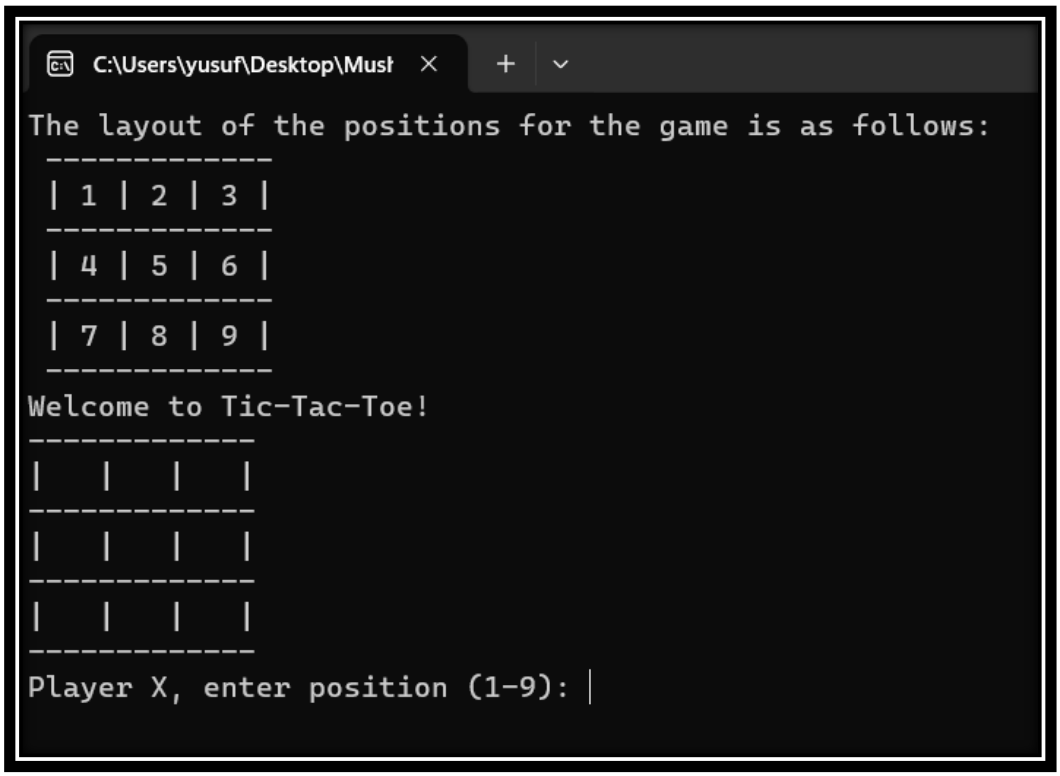
The **"main"** function serves as the central orchestrator of the game in the code. It begins by displaying a visual guide of the board positions, facilitating player understanding of the game layout. A warm welcome message follows, signaling the initiation of the game. To set the stage, the **"initializeBoard"** function is called to fill the game board with empty spaces. Subsequently, the **"playGame"** function is invoked, initiating the main game loop.

Within this loop, the current state of the board is continually displayed, and players take turns entering their moves. User input is solicited for the desired position (1-9), with input validation ensuring that the chosen position is both within the valid range and unoccupied. If an invalid move is attempted, players are prompted to retry. The game loop persists until a player secures victory or the game concludes in a draw.

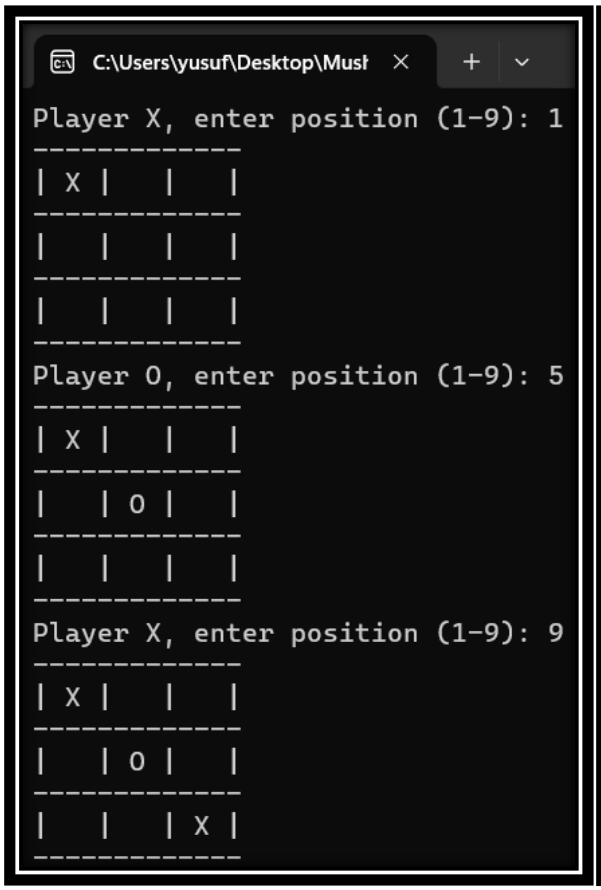
Upon each valid move, the code assesses whether the game is over by invoking the **"gameOver"** function. If a player emerges victorious, the final board state is presented, and the winning player is announced. In the event of a draw, the board is displayed, and a draw message is communicated. The **"main"** function concludes by returning **0**, denoting successful program execution.

In essence, the **"main"** function serves as the entry point and coordinator, guiding players through the Tic-Tac-Toe experience by managing the game loop, validating user input, and ultimately determining and communicating the outcome of the game.

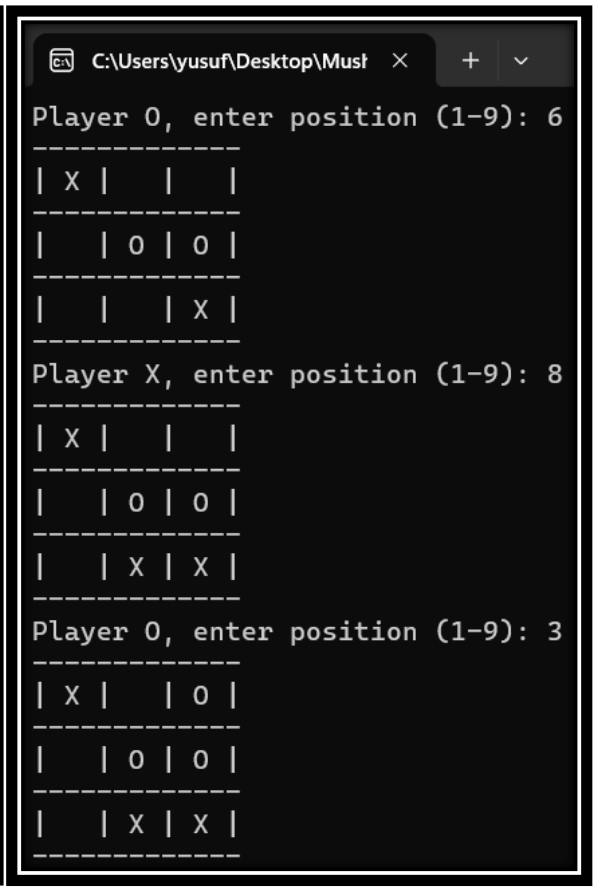
## ***OUTPUT***



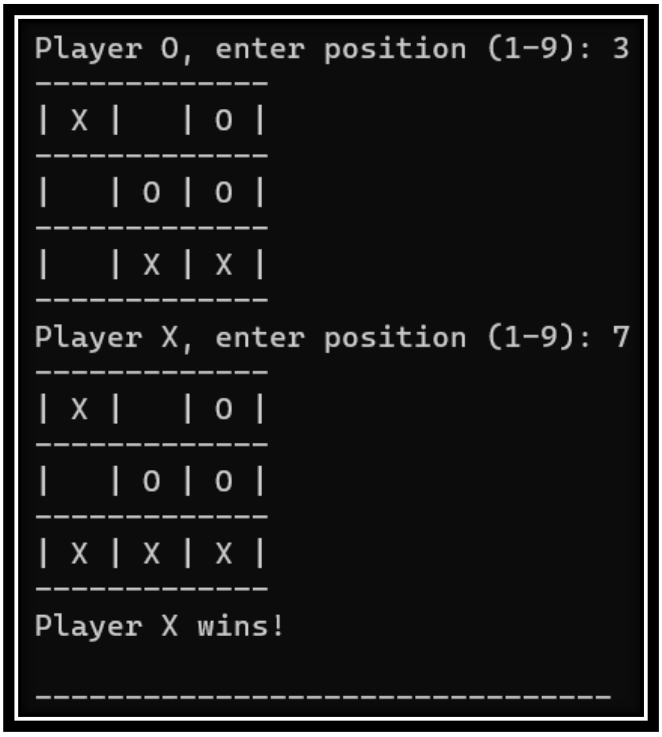

***Welcomes the players and displays the game layout***




***Output after first 3 moves***




***Output after first 6 moves***




***Player "X" wins after 7 moves***

## ***CONCLUSION***

In conclusion, the provided code encapsulates the essence of the classic game, offering a console-based and interactive experience for two players. The code elegantly structures the gameplay through functions for board initialization, display, player turns, and end-of-game checks. The main loop efficiently manages the flow of the game, displaying the board after each move and ensuring valid player input.

The incorporation of a visual guide to board positions enhances user understanding, and a welcoming message sets a friendly tone for players. The game's logic encompasses win conditions in rows, columns, and diagonals, along with a provision for detecting draws. The code handles user input gracefully, prompting players to re-enter positions in the case of invalid moves.

Overall, this Tic-Tac-Toe implementation provides a concise and effective representation of the game, combining simplicity with functionality. Whether a player secures victory or the game ends in a draw, the code communicates the outcome clearly, offering a satisfying and engaging gaming experience.