TIC TAC TOE Game with C

Introduction

The Tic Tac Toe Game is a console-based application designed with C programming language and follows the basic rules of any traditional Tic Tac Toe (XO) game. This project aims to be user-friendly and offers a variety of functionalities that you would want from a game of Tic Tac Toe. It allows playing locally against a friend or against a computer with calculated actions, however playing against the latter doesn't count in increasing your score. Playing with friends is your only option to do so and earn the title of the highest scorer as well as a great nickname. For now I'm (the creator) still the highest scorer. Do you think you can beat me?

Objectives

- Develop a user-friendly interface with clear mechanics.
- Give the player a challenging experience when faced with the computer mode.
- Implement an addicting aspect of the game (Scoring the highest).
- Ensure a satisfying experience by avoiding visual clutter (Clearing the console after each step).
- Ensure a smooth-running game, free of bugs.

Technologies Used

- **Programming language:** C
- **IDE:** CodeBlocks
- **Platforms:** Google Drive (Saving files)

  Google classroom (Submitting the project)

  ChatGPT (Solving code errors and providing a report model to follow)

Features

- **User Registration:** Players are expected to create usernames while playing against friends, inputting the same username at different times will still make the win count go up.
- **Multiple game modes:** Players have the choice to go against a local player or a computer, which will allow them to experience the game alone, or with their friends.

- **Help Menu:** Implemented in order to familiarize the players with the game mechanics and rules.
- **View Highest score:** Players can view the leaderboard to see who is in first place as well as their win count for those who are aiming for the title.
- **Play as X or O:** This option was only implemented in the play against computer mode since playing against a friend means that you will already know which sign you'll have while registrating; Player 1 → X , Player 2 →O.

Future Enhancements

- **Introducing new language options:** I thought of making the option to switch languages like french, english, arabic, Russian, Chinese. However, it seemed like too much work.
- **Enhanced leaderboard:** Add the second and the third person with the most wins in game.
- **Implement nicknames:** While I have given a nickname to the winner, I wanted to give everyone nicknames based on their wins to be called with while in game. Example: (its Yasmine THE MASTER's turn).
- **Easier re-matches:** When players ask for a rematch, the current version of the game makes players redo the registration process. I think it would be better to ask if they would like to replay with their current usernames.

Development process

The code is divided into modules:

void gotoxy(int x, int y);

I didn't use this function at first because I didn't read the guide you provided until after I finished (That's why many function are in need of optimization) but I tried it and I didn't like it (too annoying, you need to keep running the code to see where the coordinates landed) that's why I only used it in the displayMenu but I used it! That is what matters.

void displayMenu();

As it name suggests it displays the main menu.

void playGame(char currentPlayer);

This function asks the player about the mode and whether he wants to be X or O in the against computer mode.

bool playAgainstComputer(char currentPlayer);

This function plays out the computer mode when the player chooses X and asks if they want a rematch.

bool playAgainstComputer2(char currentPlayer);

This function plays out the computer mode when the player chooses O and asks if they want a rematch.

void playerMove(char currentPlayer);

This function puts either X or O (alternates) in the corresponding case when the user is playing against a friend.

void playerMoveX();

This function puts X in the corresponding case when the user is playing a against computer (O).

void playerMoveO();

This function puts O in the corresponding case when the user is playing a against computer (X).

void computerMoveX();

This function basically lets the computer put X in a random spot in its first move then checks for potential winning moves from either the player or itself and acts accordingly. It's used when the player chooses to be O in game.

void computerMoveO();

This function basically lets the computer put X in a random spot in its first move then checks for potential winning moves from either the player or itself and acts accordingly. It's used when the player chooses to be X in game.

void playAgainstFriend();

This function asks both players for their usernames and alternates between their turns then asks if they want a rematch.

void helpMenu();

As the name suggests this function prints out a set of rules and instructions on how to play.

void displayBoard();

Displays the XO board on which the game will proceed.

void resetBoard();

Resets the board's values.

void saveUsername(const char *username);

Saves the usernames to a file named usernames.txt.

bool check(const char *username);

Checks if the username is already in use.

int freespaces();

Checks the number of free spaces left in order to act accordingly.

char checkWinner();

Checks if there's a winner.

void printWinner(char winner, char currentPlayer);

Prints out the winner when the player chooses to play as X.

void printWinnerO(char winner, char currentPlayer);

Prints out the winner when the player chooses to play as O.

void viewHighestScore();

Checks who has the highest score by counting the number of wins for each username.

P.S: I am aware that some of these functions could have been optimized but as a great man (saif) has once said "If it works don't touch it".

Development Time

2 weeks for approximately 900 lines (sorry)

Conclusion

This project was a lot of fun and I (personally) consider it a success since I'm not that good at programming. I learned a lot of stuff and it's honestly easier than most exercises we do in class.

My favorite part of this project was developing algorithms for an AI player to make strategic decisions based on the current game state.

And my least favorite was the testing and debugging process.