Problem Definition

This is a Tic Tac Toe program in which a player plays against a computer in games of Tic Tac Toe. The user is allowed to play until ten games have been played or the user decides to no longer play. The overall winner is decided by which player won with the fewest turns with the computer winning ties.

Analysis

|  |  |
| --- | --- |
| **Header** | **Why** |
| #include <iostream> | Needed for I/O using std::in and std::out |
| #include <random> | Needed to generate random numbers for the computers turn |

Variables

|  |  |
| --- | --- |
| **Name** | **Use** |
| board | An array used to keep track of the player and computer moves on the tic tac toe board |
| score | An array used to keep track of the wins each player gets as well as how many turns it took for that win |
| stillPlaying | The variable checked to continue or stop the programs execution |
| gameWon | The variable checked to see if an individual game of tic tac toe was won |
| numberOfGamesPlayed | Keeps track of the total number of games played to make sure no more than ten games are played (filling the scores array beyond its set size) |
| turnCount | Keeps track of the total turns each player makes to determine who wins overall after all tic tac toe games |
| gameWon | Keeps track of whether or not a player has won the game, and if they do end the current game |
| row | Keeps track of the current row being worked with |
| col | Keeps track of the current col being worked with |
| validTurn | Keeps track of whether a particular move made by a player is legal |
| input | Takes the user input for if they would like to keep playing |
| winner | Keeps track of the current player whose win took the fewest terns |
| playerWins | Keeps track of the total player wins |
| computerWins | Keeps track of the total computer wins |
| catWins | Keeps track of the total cat wins |
| WINNERS | A global constant that clarifies what the index for the score’s column is for |
| TURNS | A global constant that clarifies what the index for the scores column is for |
| ROWS | A global constant that stands in for the total number of rows on the board |
| COLS | A global constant that stands in for the total number of columns on the board |
| MAXGAMESPERSESSION | A global constant that stands in for the maximum number of games that can be played in a session |
| PLAYER | A global constant that stands in for an integer representing the players wins |
| COMPUTER | A global constant that stands in for an integer representing the computers wins |
| CAT | A global constant that stands in for an integer representing the cats’ wins |
| COL1 | A global constant that stands in for the index of column one |
| COL2 | A global constant that stands in for the index of column two |
| COL3 | A global constant that stands in for the index of column three |
| ROW1 | A global constant that stands in for the index of row one |
| ROW2 | A global constant that stands in for the index of row two |
| ROW3 | A global constant that stands in for the index of row three |

Design

See pseudocode word attachment.

. I tested my program by playing different combinations of moves during the games of Tic Tac Toe, making sure to rotate who was winning between the user, the computer, and the cat. These were some of the issues I ran into.

1. would allow a player to win even if they didn't have 3 in a row.

--- FIX: comparison error, only one == 'X'

1. Would get stuck in an infinite loop if it was the last possible move on the board.

--- FIX: the still playing loop made gameWon false making the intro message repeat infinitely.

1. Second diagonal was not reporting wins when it should have been.

--- FIX: I had defined two winning conditions for the first diagonal and none for the second.

1. cin was buffering inputs when getting the players input which would cause some very strange integer values to fill the row and col variables.

--- FIX: Use cin.clear to remove the input error and then cin.ignore to clear the buffer after each input.