Problem Definition

This is a program that will a human user to play a game of tic-tac-toe against a computer opponent. The board marker locations should be stored in a 3x3 array that will be initialized to a whitespace character. These array locations should be updated using the player_turn and computer_turn functions that allow the user and computer to choose marker locations. Score should be kept in a 2x10 matrix. The check_winner function will be called each turn to determine if a winner was found. Each round, the user will be asked if they want to play again, this will continue as long as the user answers yes, or the scoreboard is full. Upon completion of all rounds, the overall winner will be selected based on minimum number of turns taken, and the winner message will be displayed, along with the minimum number of turns taken to win.

Analysis

Header	Why
#include <iostream>	Needed to handle Input\Output
#include <cstdlib>	Needed for the rand() function
#include <ctime>	Needed to seed the srand() function

Variables in main()

Name	Use
PLAYER_MARK	Constant char used to hold ‘X’
COMPUTER_MARK	Constant char used to hold ‘O’
the_board	3x3 char array used to hold marker locations Initialized to whitespace ‘ ‘.
play_again	char variable used to hold users ‘y’ or ‘n’ answer .
score_board	Integer array used to keep track of who won, and how many turns each round took. Initialized to 0, using the number 1 to mark a player win and number 2 to mark a computer wi in the first column.

rounds_played	An accumulator to keep track of how many rounds have been played. Initialized to zero.
player_turns	An accumulator used to keep track of turns for player within a round.
cpu_turns	An accumulator used to keep track of turns for computer within a round.

Variables in initialize()

Name	Use
board	Parameter for 3x3 array that contains marker locations.
row	Integer used in nested for loop to iterate over each row value.
column	Integer used in nested for loop to iterate over each column value.

Variables in draw_board()

Name	Use
board	Parameter for 3x3 array that contains marker locations.
row	Integer used in nested for loop to iterate over each row value.
column	Integer used in nested for loop to iterate over each column value.

Variables in player_turn()

Name	Use
board	Parameter for 3x3 array that contains marker locations.
mark	char Parameter that will hold ‘X’.
turn_done	Bool flag variable used to exit while loop ones the users turn is complete. Initialized to false.

row	Integer used to hold players choice for row location.
column	Integer used used to hold players choice for column location.

Variables incomputer_turn()

Name	Use
board	Parameter for 3x3 array that contains marker locations.
cpu_mark	char Parameter that will hold ‘O’.
row	Integer used to hold randomly generated number in the range 0 to 2.
column	Integer used to hold randomly generated number in the range 0 to 2.

Variables check_winner()

Name	Use
board	Parameter for 3x3 array that contains marker locations.
mark	char Parameter that will hold either ‘X’ or ‘O’.
result	Bool variable that will be set to true if a winning pattern is found. Initialized to false

Variables display_overall_winner()

Name	Use
score_board	2x10 Integer parameter used to hold the score_board array
rounds	Integer parameter used as a limit on the amount of columns iterated over in the score_board array
player_wins	Integer accumulator used to hold total wins the player has achieved. Initialized to 0
computer_wins	Integer accumulator to hold total wins the computer has achieved. Initialized to 0

min_player_turns	Integer used to hold the minimum number of turns taken for the player to win a round. Initialized to 9
min_computer_turns	Integer used to hold the minimum number of turns taken for the computer to win a round. Initialized to 9
column	Integer used in for loop to iterate over columns in score_board array.

Design

Algorithm

1.Do_while loop:

1. Intialize turns to 0
2. While loop
    1. Initialize board to whitespace
    2. Draw current board
    3. Check for winning patterns and rounds in range
    4. Determine whos turn it is
    5. Execute player\computers turn
    6. Draw current board
    7. Increment turns by 1
3. Determine winner for current round
4. Display message to the user saying who won the current round
5. Record winner and turns taken to win in first and second row of scoreboard array
6. Increment rounds played by 1
7. Check if rounds played is less than scoreboard size
8. If rounds played is less than scoreboard size, ask user if they want to play again, otherwise set play_again equal to ‘n’ and exit the do while loop because the scoreboard is full.

2. Once the game is over, calculate total wins for each player , and minimum number of turns taken by each player, and display that information to the console

Implementation

This program was written using VS Community edition, using a Windows 7 computer running an Intel core i-5 4690k with 16 gigabytes of ram. Upon initial testing, I had an issue with the computer accepting input for row and column locations during player turn that was out of range, this was corrected by adding a range check to the player turn function. Another issue I had during testing was the program crashing if the scoreboard was full and the user continued to play the game, this was corrected by adding a rounds_played check to the do- while loop. An alternative is to add columns to the scoreboard increasing the amount of slots for winners and scores, but this seems unnecessary for this case. The program currently appears to be operating smoothly, with all values being what I anticipated them to be by completion of game.