William George

11/16/2014

CS 161

Week 7 Project

UNDERSTANDING

The purpose of this weeks’ project and exercises is to practice with 1 Dimensional and 2 Dimensional arrays as well as practicing with command line inputs, and dynamic arrays. It looks to be challenging as these are not things we’ve used before and 2D arrays use a lot of nested loops, which is not something we’ve had to use a lot of so far in this course.

The program for the main project will take a user’s command line input for the number of games of tic tac toe the user would want to play. If there is no input then the program will just play a single game. The program will then begin with a two player game on a 3x3 board that will display whose turn it is have the user input their choice based on the coordinates and output a new board with the users input included. The program will alternate player turns until a player either wins or the board is filled. If there is another game left, or if there is not an overall winner then the other player will get to start the next turn. Each time a player gets three in a row, column or diagonal then the program will output whether the player won and break from the game. At the end if a player has won enough games to be the overall winner then the program will tell them they are the winner and end.

This program will be a challenge as it is much more involved than many of our previous programs. I feel like we will have a much longer program with many more functions than anything we have built previously.

DESIGN

Preprocess Directives:

Iostream, Cstdlib

Global variables:

int COLUMNS = 4,int ROWS =4

Functions:

Winner function

PlayerChoice function(s)

Coordinates valid function

New board function

MAIN Program

Variables :

numGames, numTurns, player x wins counter, player o wins counter

gameboard 2d array

GET command line input for numGames

IF command line input is not input play one game

ELSE numGames = command line input

WHILE numGames does not = 0 play game

OUTPUT new game board – CALL new gameBoard function

IF numGames is even player X starts

WHILE numTurns not = 9

CALL PlayerX choice function

IF PlayerX wins

Player X wins counter ++

BREAK

IF num turns = 9

BREAK

CALL PlayerO choice function

IF PlayerO wins

Player O wins counter++

BREAK

ELSE player O starts

WHILE numTurn not= 9

CALL PlayerO choice function

IF PlayerO wins

Player O wins counter++

BREAK

IF num turns = 9

BREAK

CALL PlayerX choice function

IF PlayerX wins

Player X wins counter ++

BREAK

IF numTurns == 9

OUTPUT “Tie Game”

OUTPUT player O and player X win counters

IF player O or player X win counters > total number of games/2

OUTPUT overall winner

BREAK

OUTPUT press enter to play again

CLEAR screen

DECREMENT numGames

END main

NEWBOARD function

Variables: cols, rows

FOR each row

FOR each column

IF row and column = 0

PLACE space at 0,0

ELSE IF row = 0

COLUMNS = column - 1

ELSE IF column = 0

ROW = row – 1

ELSE

PLACE ‘.’ in space

OUTPUT gameBoard

END Function

PLAYER X CHOICE FUNCTION/PLAYER O CHOICE FUNCTIONS

Variables :

SET int x, y, row, col, coordinate

SET choice array

OUTPUT: Player X,O enter move

FOR 2 INPUTS

INPUT coordinates to fill array

WHILE choice is invalid OR space is taken

OUTPUT enter valid choice

CLEAR buffer

INPUT new choice

SET choice to x,y

PLACE X or O in array at user choice

OUTPUT updated gameBoard

END function

BOOLEAN WINNER Function

Variables: row, col

FOR each row

IF every column = X OR O

RETURN true

FOR each column

IF every row = X or O

RETURN true

IF Left diagonal = X or O

RETURN true

IF right diagonal = X or O

RETURN true

RETURN false

END FUNCTION

BOOLEAN isNum validation input

IF numInput doesn’t = 0 OR 1 OR 2

RETURN false

RETURN true

TESTING

|  |  |  |  |
| --- | --- | --- | --- |
| INPUT | EXPECTED OUTPUT | ACTUAL OUTPUT | NOTES |
| ticTacToe 3 | You will be playing 3 games | Expected |  |
| 0 0 | 0 1 2  0 o  1  2 | Expected |  |
| 0 1 | 0 1 2  0 o x  1  2 | Expected |  |
| 0 2 | 0 1 2  0 o x o  1  2 | Expected |  |
| 0 3 | Not a valid input | Expected |  |
| 1 0 | 0 1 2  0 o x o  1 x  2 | Expected |  |
| 1 1 | 0 1 2  0 o x o  1 x o  2 | Expected |  |
| 1 2 | 0 1 2  0 o x o  1 x o x  2 | Expected |  |
| 2 1 | 0 1 2  0 o x o  1 x o x  2 o | Expected |  |
| 2 1 | There is already a selection in that space | Expected |  |
| 2 0 | 0 1 2  0 o x o  1 x o x  2 x o o | Expected |  |

REFLECTION

The main project was especially tough this week. Not necessarily because the logic was difficult, because once I got the project on paper it all came together quickly, but because of the amount of code needed to get a working project. The most frustrating part of the project was outputting the new game board actually. I didn’t realize how difficult it would be to input an integer into a character array and eventually had to give up and just put individual characters into each row or column.

I also think my code may be a bit more inefficient than what would be ideal, but because of time constraints I don’t want to mess with it anymore. Specifically I believe there is a way of putting each of the player choice functions into one and I believe there should be some way of validating the coordinate input via function that would not make me check the input twice for the correct number input when the user first inputs the number and again when the program checks the function to see if the space is filled and has to ask for a number input from the user again.