



CIS 345 – Business Information Systems Development II – Fall 2014

Assignment 4

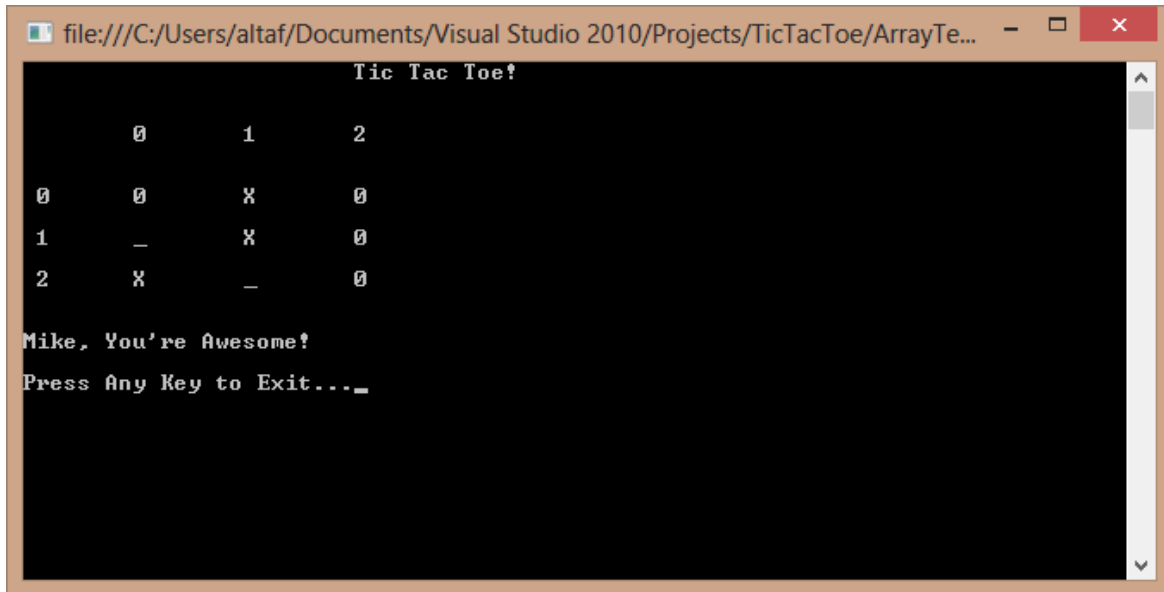
Due on Blackboard on Friday, October 3, at 10 PM

This assignment requires you to complete an unfinished application that implements the Tic Tac Toe game.

Software developers frequently find themselves working on programs written by others, making modifications to existing code (with no comments!), or finishing programs started by others. Another frequently encountered situation is writing internal method logic while being provided with a method structure or interface (i.e. parameters, output type). This assignment simulates those situations. One of your primary goals for this assignment should be to fully understand the working of the program logic.

Read and understand the intended behavior and setup of the program thoroughly before starting to work on the application.

Skills developed: Nested loops, utilizing multidimensional arrays, separating data storage from data presentation, developing, understanding, and adapting moderately sophisticated logical structures.



```
file:///C:/Users/altaf/Documents/Visual Studio 2010/Projects/TicTacToe/ArrayTe...
Tic Tac Toe?

      0      1      2
0      0      X      0
1      -      X      0
2      X      -      0

Mike, You're Awesome!
Press Any Key to Exit..._
```

Implementing Tic Tac Toe

Storing Data

A Tic Tac Toe game involves 2 players placing "0" or "X" marks on a 3*3 matrix. This program will simulate a Tic Tac Toe board by having a 2 dimensional array. The data in the array will consist of integers. Player 0's moves will be indicated by 0. Player 1's moves will be indicated by 1. Blank cells that have not been used for moves yet will contain -1.

Since an integer array contains 0s at initialization, a reset method will be called, that places -1 in all elements at the start of the game. This structure is better than putting in -1s at array initialization because it allows room for calling the reset method again. In the future, if one were to enhance this application to start a new game, this method could be utilized for that purpose.

The user will not see "-1" or "1" even though that is how the data is stored in the array. A method called ToXO() will take a number (0, 1, or -1) and return a string to display to the user (O, X, or underscore respectively). This illustrates how data storage can be made different than data presentation. *The user's need to see data in a certain format doesn't mean that you must store it in that same format. Store data the way you want it stored and then convert it when it needs to be displayed. Storage and presentation are two completely different things!*

Establishing an Underlying Logical Structure

The player numbers (0 for the first player and 1 for the second player) will be used for multiple purposes. 0 and 1 are nameArray index locations. 0 and 1 are player numbers stored in currentPlayer. 0 and 1 are values of elements that store the Tic Tac Toe data regarding moves. This structure allows the currentPlayer variable to be utilized uniformly across the application for everything pertaining to either of the players. The value in the variable seamlessly adjusts the program to accommodate change of players.

Assessing Wins

After each user move, the program needs to check if the user has won or not. There are 8 ways a user can win. He or she can complete any one of the 3 rows, or the 3 columns, or the 2 diagonals. There is one method to assess the completion of rows, one method to assess the columns, and one method each for both of the diagonals. All of these methods are called by TestForWin(). If the game has been won, as determined by these methods, a boolean value, gameWon is set to true and the program terminates. Otherwise, the program keeps running.

Bad values input by the user such as an invalid cell locations or a cell location that has already been used will also terminate the program.

Assignment Requirements

Your task is to complete portions of the code as specified below. You **MUST** use the variable and method names specified below and in line with those used in the program. **The variable/method names cannot change by even one character.**

Download, uncompress, and utilize the project available on Blackboard. Fill in code within the provided project. After completion, zip that same project and submit.

Further implementation details regarding each of the steps are provided inline within the program as comments.

1. Declare a static array of strings in the class and allocate it with a capacity of 3 elements. It should be called nameArray.
2. Declare a static 2-dimensional array of integers in the class and initialize it with 3 rows and 3 columns. It should be called gameMatrix.
3. In Main():
 - 3.1 Write a set of three calls within the Main method before the do loop.
 - 3.2 Write a set of calls within the do loop.
4. Complete contents of ResetGame(). It loops through the full 2-dimensional array and sets every element to -1.
5. Complete contents of GameWonByRows() by looking at GameWonByCols(). *Rather than copying and pasting the entire contents, type it out yourself so as to maximize your learning.*
6. Complete the code inside the inner loop in
 - 6.1 GameWonByDiag1()
 - 6.2 GameWonByDiag2()
7. Write 2 lines of code in TestForWin()
 - 7.1 One line for determining if either of the diagonals has been completed.
 - 7.2 One line for determining if the overall game has been won
8. In, PlayNextMove(), complete the if block
 - 8.1 Fill in the condition within the parentheses
 - 8.2 Write one line to assign a value to an element within the gameMatrix using the row and column numbers.
9. Complete the contents of ToXO(). It returns a string which shows what to display to the user. An input of 0 returns "O", 1 returns "X", and -1 returns "_".

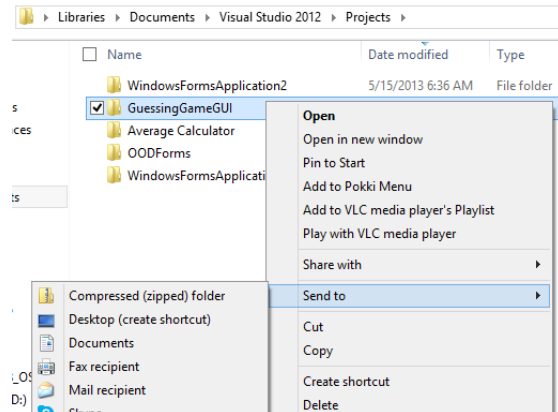
Optional Learning Challenges (Do not submit these with your assignment).

- Can you integrate the two methods for computing diagonal wins into one method and integrate the two methods for computing row/column wins into one method? Look into passing an additional parameter value (0/1), which tells the method what needs to be evaluated.
- Developing logic for winning the game is one of the primary challenges of implementing this program. Without looking at how the four methods to determine wins have been implemented in the template, write your own methods to determine if a user has won.
- Increase the matrix size and adjust the program to accommodate the increased size.
- Setting up a data storage structure and storing the values into that structure is the other primary challenge in implementing the program. Modify the application to utilize a 2-dimensional array of strings that stores "O" "X" or "_". Currently the currentPlayer variable acts as the value of the cell - that would have to change in the alternate structure. *This is a fairly comprehensive change since it alters the underlying structure of data storage.*
- Think about what it would require to make the game 3 dimensional. Can you implement it? (There are 3D arrays available in C# - implemented slightly differently) *This is fairly complex. Even if you don't implement it in C#, work out the logic of finding out if a player has won. That means determining whether a row, column, or diagonal has been completed in any of the X, Y, or Z dimensions. Remember that most of the challenge is working out the logic!*
 - If you HAD TO display this on a 2D console screen, how might you manage it?
- Create the 2D game from scratch! :-) *It's more challenging to do it in a console environment than it would have been with a GUI!*

Submission Instructions

Submission should be made using a zip file that contains all of the Visual Studio C# project files. You will need to **zip the entire project folder** along with the .sln and .suo files. The folder will automatically contain the class source files as well as the executable file that is generated in \ProjectName\bin\Debug folder. Upload file to the Blackboard assignment drop box.

Zip the entire top-level folder by right-clicking the folder and selecting Send to | Compressed (zipped) folder.



Using built-in windows zip tools: <http://windows.microsoft.com/en-us/windows/compress-uncompress-files-zip-files>

Make sure you check the following. Your grade is dependent on all these criteria being met.

- You have included your name as a comment within your class.
 - e.g. “// Assignment 4, Jane C. Smith, CIS 345, Tuesday 9:00 AM”
- Zip filename is: FirstNameLastName_Assignment4.zip
- Your code is commented and you are using all prescribed programming conventions.
- Your code utilizes PascalCase and camelCase as appropriate.

Verify your zip file before you submit

- Check for actual class files being present in the folder before you zip it.
- Check your zip file size after zipping – if it is 1K, it likely contains only a shortcut.
- Uncompress your zip file before submitting and verify that files are present.
- Download your zip file after submitting, uncompress, and again verify that your files are present. Test your files in Visual Studio after uncompressing.
- **Make sure you have submitted your file and not just saved a draft on Blackboard. A blue clock indicates a submission in progress, i.e. a draft, not a submission.**

This takes an extra couple of minutes. Please do it if your grade is important to you. If you do this, you will not end up submitting a bad file. If you submit an empty file, or one containing only a shortcut, or a bad zip file, or a bad project file, you will receive a score of zero and your only recourse will be to do the makeup assignment at the end of the semester.