Daniel Domalik
CIS3190
Michael Wirth
02/02/2018

Assignment 1, Fortran Tic Tac Toe

To begin, I will describe in short how the program functions. When you run the program, it will display the game board with numbers that help the user understand how to place their “X” in the proper place. The game then asks for the users input, as soon as the user inputs a number the program checks if it is a valid number from 1-9, if it is, it checks if the spot is available. If the spot is valid, the user’s input is allowed, otherwise the user is prompted to enter another number. The AI then places an input based on the algorithm given in the specification. If no sum of numbers is found to win the game or block the opponent from winning the game, the AI places their move into a random spot. After every input, from the user or AI, a check if performed to see if the game is over or not. If the game is over, the program breaks out of the DO loop, prints a message, and ends the game.

The first thing I tackled after reading the specification is what kind of data structure I could use. I took a look at the given subroutines and established that a character array would work just fine. A 9 character long array that I can pass in to all my functions if necessary. The next thing I did was read through the given subroutines and try to understand what they were trying to accomplish. After understanding what they were made for, I started to change them to fit a more modern Fortran coding convention. This included things like removing the GO TO statements, changing the variable conventions, and converting the logical/relational operators. The CKPLAY subroutine was changed from a 2D array to a 1D array, and the logical/relational operators were changed to fit new Fortran coding conventions. The CHKOVR subroutine was changed from a subroutine that handled a 2D array, to one that handles a 1D array. Instead of accessing the third cell at “tictac(1,3)”, it would just be “tictac(3)”. This made the code simpler and easier to understand. I used the same logic from the subroutine but modified it to work with the new array. In my opinion, there was no reason to use the 2D array, when the 1D array would accomplish the end task without any issue.

Understanding the code itself was no problem for me. My experience with VB (and programming in general) helped lots, due to the similarities when creating loops and such. Fortran is not that big of step away from the languages I usually use. It is easy to understand and did not have any difficulties understand any of the code or writing new code. Of course, I did a little reading on syntax, but other than that it was fairly simple. However, this being the first assignment, I expect it to get harder and more redundant as we go deeper into the older languages.

Being able to write proper Fortran code, I dove in. After re-engineering the given subroutines, I started with user input. A couple of WRITE statements and READ statements, and I had my board as well as the user being able to input a move. Moving from there, I did defensive programming on the user input so that anything other than an integer between 1-9 was not allowed. I cited a solution to check if the input was an integer or not, checking if it was between 1 and 9 was done using logical/relational operators. After I had finished the user input and game board display, I moved on to the bulk of the program, the AI. I followed the specification to program an algorithm for the AI. It checks all 8 paths, row by row, column by column, and the two diagonals. If a sum of 8 is found in one of these paths, the AI places another “O” in the empty space and wins the game. It then checks for a sum of 2, if it is found the AI places an “O” in the missing space to block the user from winning the game. If neither of these sums are found, the AI places an “O” in a random spot on the game board.

After having finished the AI, I moved on to integrating the provided subroutines into the program. The first subroutine I added in was the CHKPLAY so that I did not have the AI possibly overwriting the users move. Adding this subroutine was a simple, “plug-and-play”. I just copy pasted the code over and it worked. The next subroutine, however, was not as simple as the first one I added in. I realized that I needed to create a new function called, “SAME”, to verify if three characters passed in were the same characters (either “X” or “O”). I accomplished this creating two IF statements. As soon as the function was accessed, the variable “SAME” was set to false. The first IF statement checked if all three characters were X’s, if they were, set same to TRUE and return it. The second IF statement did the same thing except checked if all three characters were O’s.

Finishing the program and reflecting, I realized that it was quite simple to create Tic-Tac-Toe in Fortran and re-engineer the given subroutines. In my opinion, it would have been easier to create this program in C. This is probably due to the fact that I have used C plenty of times in the past so I am more comfortable with it. One feature I noticed in Fortran that was useful was the check if an input was an integer. It was one simple line and was integrated into the actual WRITE statement. In C you would have to parse the input first, and then check it against some logic in an IF statement, perhaps.

All in all, I enjoyed my experience with Fortran. Building this simple program made me realize that Fortran isn’t as bad as I may have portrayed it to be. I thought it because it is an older language I would have trouble with it, but then thought about how old C is and everything sort of fell into place. It was similar to languages I have used in the past and nothing was too hard to understand or code. I look forward to the second assignment and what Ada is like.