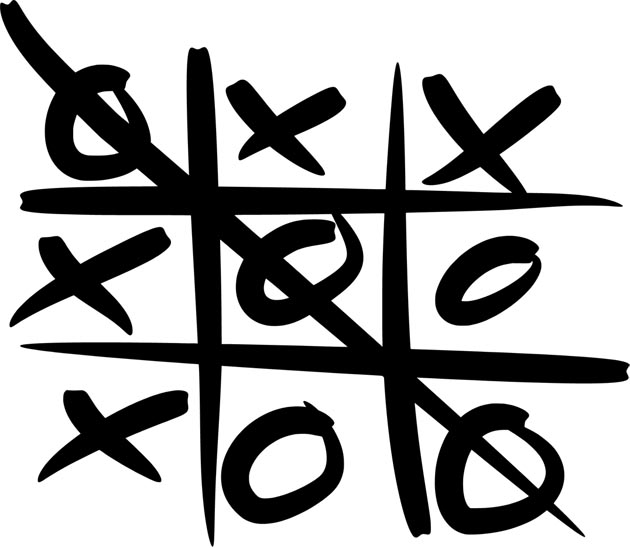
**PROJECT 1**

TIC TAC TOE



Anh Vu

CSC 5 – 46023

July 18th, 2014

**TABLE OF CONTENTS**

**Cover**……………………………………………………………………………………….…….1

**Introduction**……………………………………………………………………………………...3

**Rules**…………………………………………………………………………………...…3

**Design Details**................................................................................................................................ 4

**Research**.........................................................................................................................................7

**INTRODUCTION**

Tic-Tac-Toe is a game in which two players, represented as either X or O, mark a 3 x 3 grid until one player wins. The objective of the game is to mark 3 spaces that are horizontally, vertically, or diagonally adjacent to one another before the other player does.

**Rules –**

* Each player will take turns marking the grid. No player will mark the grid more than once at a time.
* You must only mark the empty spaces in the grid.
* Many times you will find that neither player wins. This game is considered a draw, or a cat’s game.

**Instructions –**

* The program will randomly decide which player (‘X’ or ‘O’) will play first
* Each space in the 3 x 3 grid will be represented by a number. Please select the numbered space in which you would like to place your mark. Your mark will be placed on the grid after each time you input.
* The game will continue to run until 1) one person wins, or 2) all spaces are occupied and no one wins.
* If you wish to see your total scores, see the text file names "scores"
* Enjoy the game!

**DESIGN DETAILS**

**Approach -**

In order to successfully code Tic-Tac-Toe, I first played a game and broke the game down in small steps in order to fully understand the sub components of the game and to ultimately implement them in a code. Below are the steps I initially drafted:

1. Draw and output a 3x3 board.
2. Designate each player as either 'X' or 'O'
3. Allow users to take turns and to mark the board with their respective marks.
4. Conditions for the game to end
5. Ways which players can win
6. Keeping scores and outputting them so user can see

**Overview -**

At the end of my project, I had developed a game that had encompassed all of the steps that I had originally thought of. (Step 1) My program outputs a board each time a player makes a move. The board I created consists of a set of variables that represents each space. An example of my board is below:

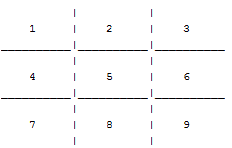


Figure 1. My tic-tac-toe board. In order to make a move, each player is required to input the number where he or she wants to place his or her mark.

If the space is already taken up, the program prompts the user to input another number. (Step 2) My program also designates Player 1 as 'X' and Player 2 as 'O' and randomly chooses which player will be able to play first. Then, the program switches between each player from one move to the next. (Step 3) When the player chooses a number to place his mark, the number in the grid will be substituted with the mark. An example is shown below:

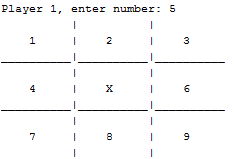


Figure 2. When player 1 (X) chooses number 5, that specific space is then replaced with player 1's mark.

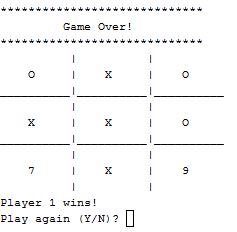
(Step 4) Of course, when playing any game there are conditions to signal the end of the game. The game can end in nine different ways. Three ways the game can is if there are the same marks adjacent to one another in each of the three horizontal rows of the board (e.g. spaces 1, 2, and 3). Another three ways the game can end is the same as the aforementioned condition, but it applies to each of the three vertical columns. Two ways that will end the game is if the same marks are adjacent to one another diagonally. The last way the game can end is if all the spaces are taken up. In this case, neither player wins and the game is considered a draw. Below shows an example of an output to signal the end of the game. Also note that I prompt the user to decide whether he or she wants to play again.

Figure 3. Player 1 wins because he/she placed 3 marks adjacent to one another vertically before the other player could. The program also asks the user if he or she wants to play again.

C:\Users\user\Desktop\Capture.PNG(Step 5) As you can see in *figure 3*, the program is able to determine who wins. The condition that determines who wins is which player is able to end the game first. (Step 6) When each game ends, the scores for each player are subsequently added depending on which player wins. Scores are tallied and output on a text file titled "scores." Below is a sample of a the text file:

Figure 4. Player 1 has one win while Player 2 has none.

The details of my program will be further discussed throughout my report.

**RESEARCH**

In order to code my game, I had to research and study the following topics that were not covered in class thus far:

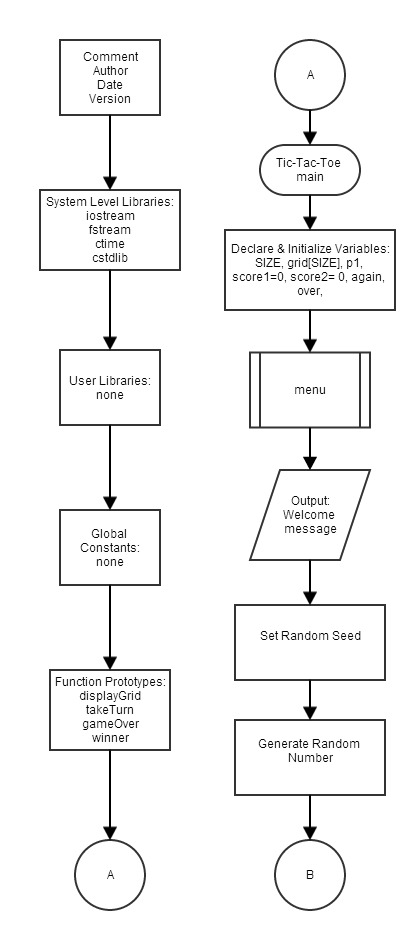
1. **Arrays -**

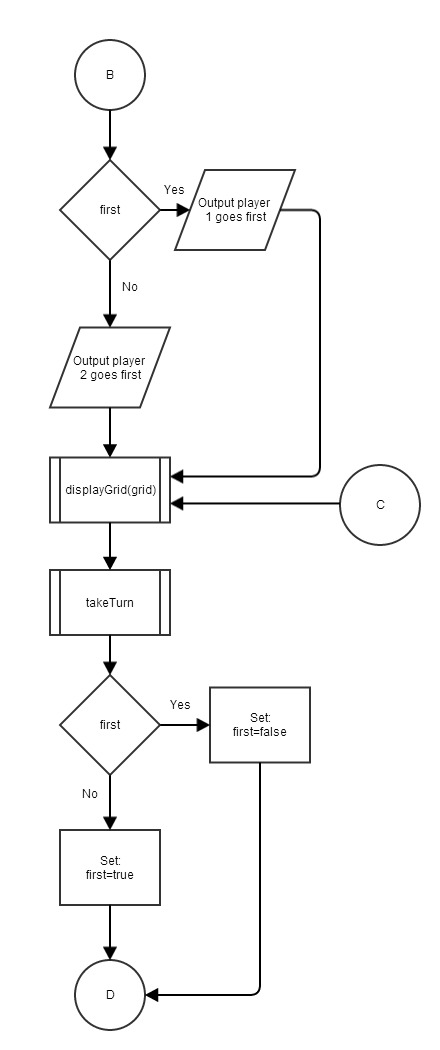
I used arrays so I could hold the variables that would represent each space on the board. When the user inputs where they want their mark, that variable in the array would be replaced with the user's respective mark. I mostly used arrays in order to hold a lot of variables on my tic-tac-toe board; this made it very easy for me to reassign each space with a mark throughout the game.

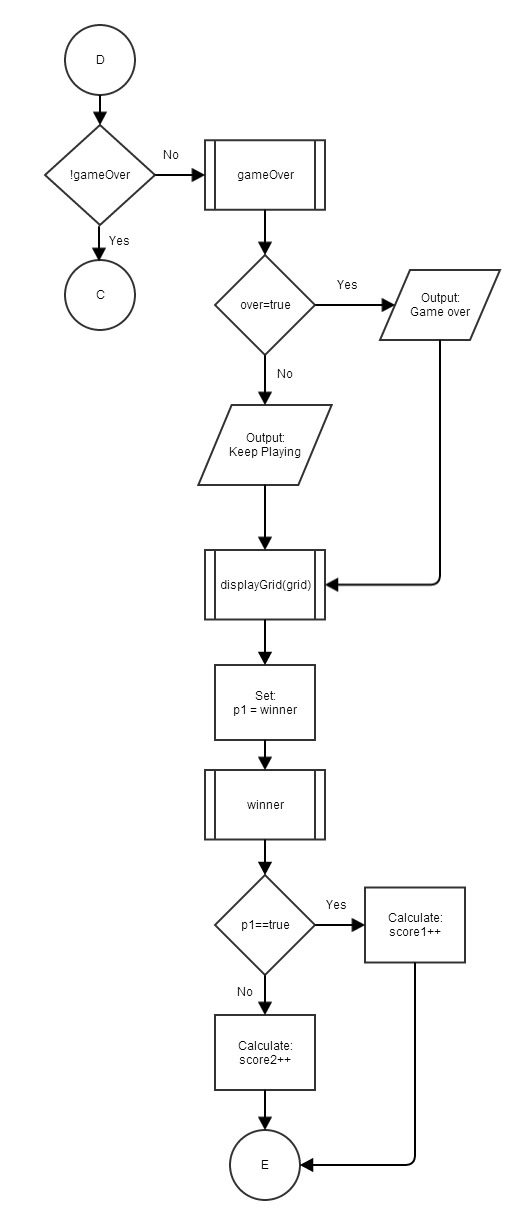
1. **Functions -**

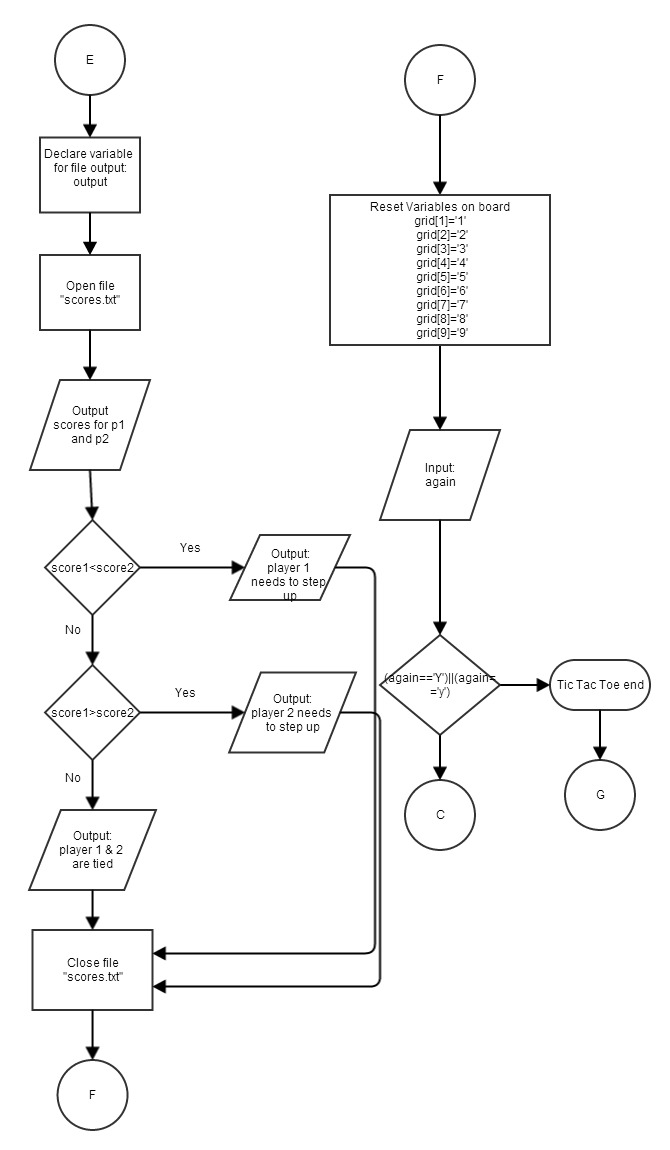
Without functions, my code would have been unnecessarily long. I would have to repeat my code to output a grid every time a player makes a move. The same applies for determining the winner or whether the game is over. Instead of repeating the same lines of code over and over again, declaring a function and defining them would be much easier. Instead of repeating lines of code, I could call functions that would perform these codes. It greatly simplified the layout of my code and made it much easier to read and understand. I have separate functions for printing out the grid, determining whether the game is over, determining the winner, and outputting the right marks on each space.

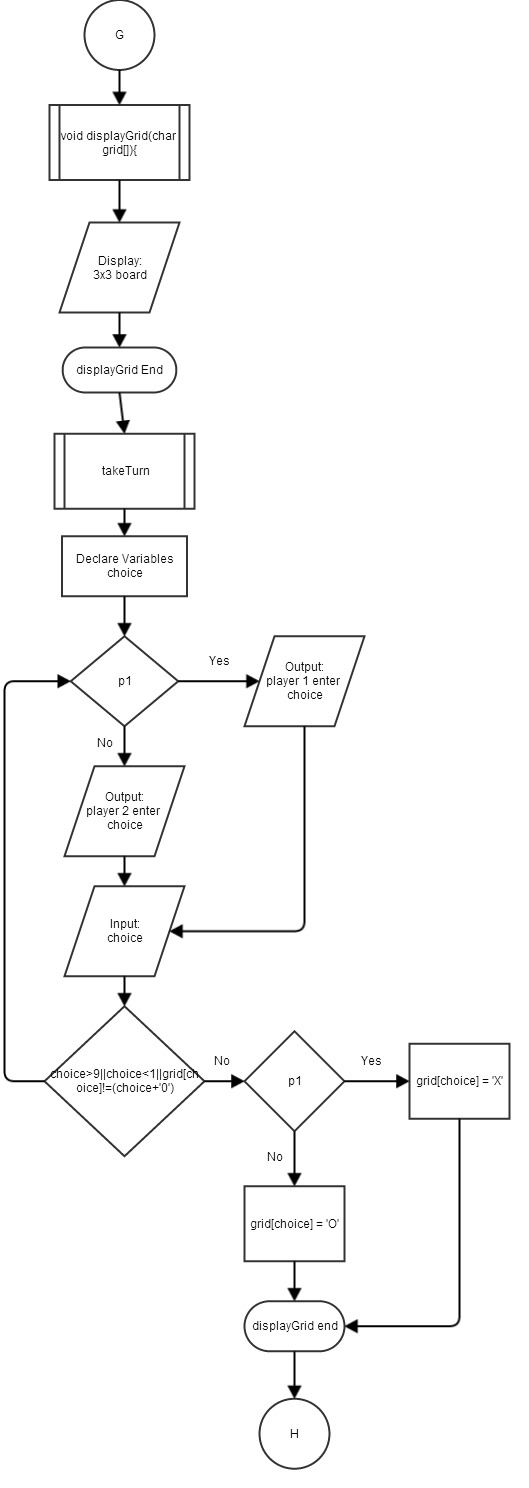
**FLOWCHART**

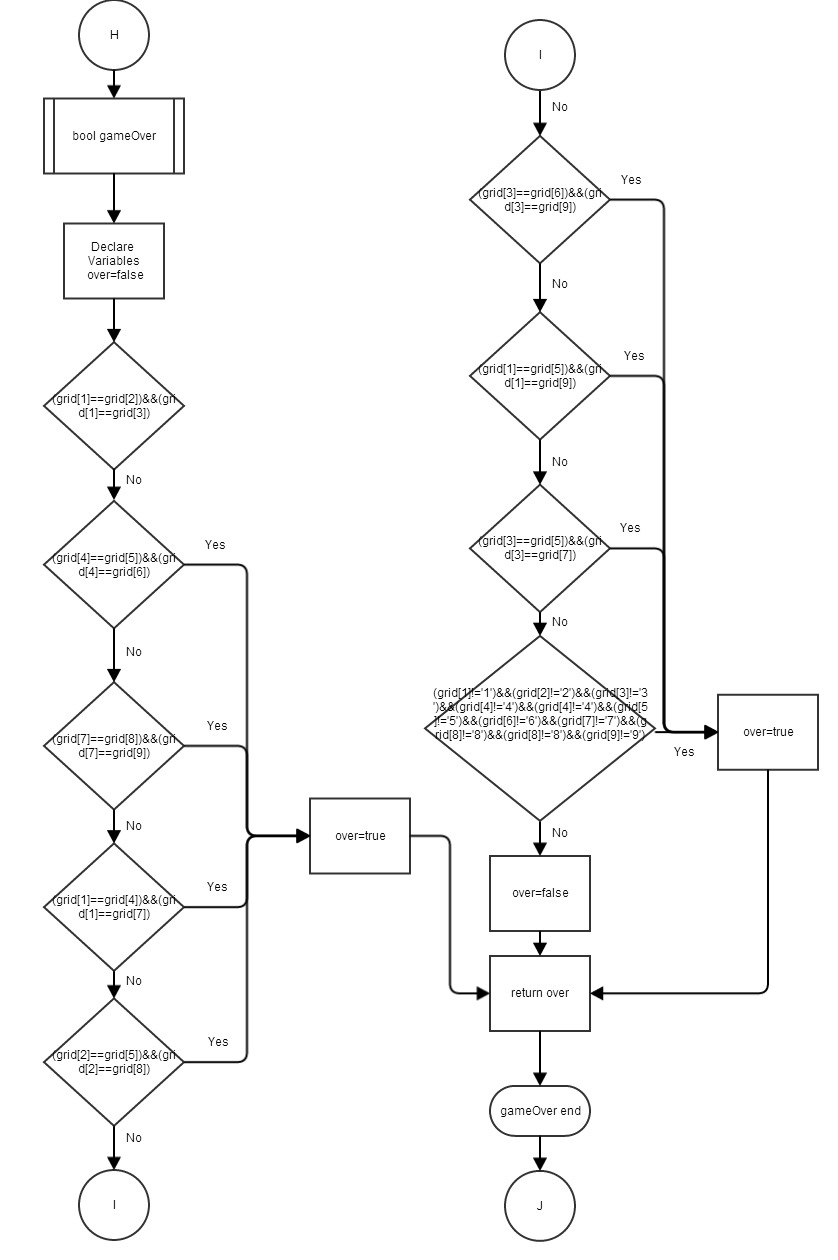
****

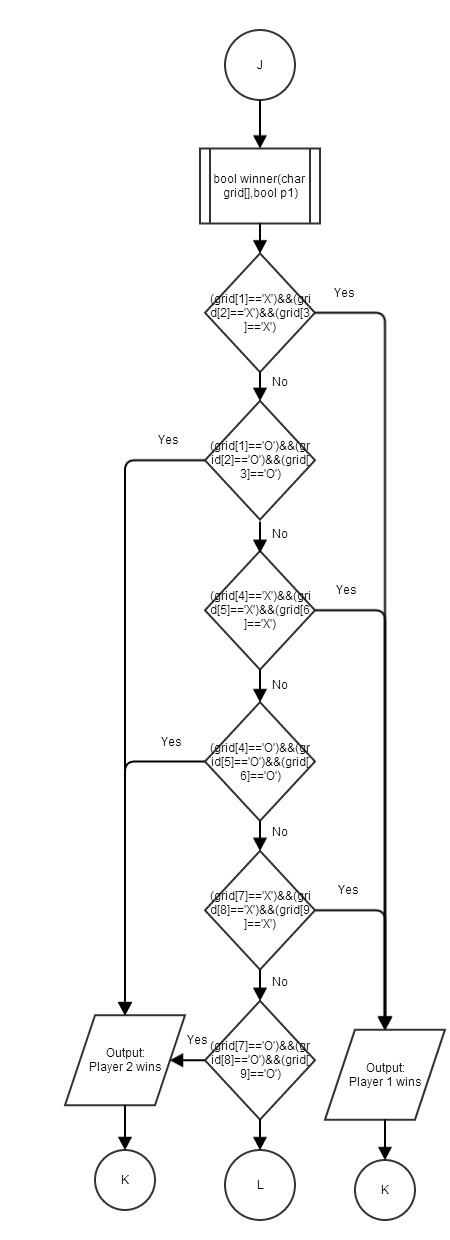
****

****

****

****

****

****

