Project 2: 2Games

Marwah Dawoody

CSC 5 - 07/28/14

Introduction:

Title: 2games

1. The first game is (was meant to be)a simple tic tac toe game
2. Rock Paper Scissors Lizard Spock, is your typical game of Rock Paper Scissors with two additional elements Lizard and Spock. The rules go as follows:

* Rock beats scissors
* Scissors beat paper
* Paper beats rock
* Lizard beats paper
* Rock beats lizard
* Scissors beat lizard
* Spock beats rock
* Lizard beats spock
* Paper beats spock

Summary:

Project Size: 200 lines

Variables:

* short plc (the player's choice in the game)
* char pet[30] and color[13] (used to create some sort of user name), cont1 and cont2 (whether or not to continue playing the game)
* int game (determine which game the player wants to play)
* int ttt[9] (sopposed to be the array for the tic tac toe game)

This is the second version of my first project. Unlike the first project this one had to implement the use of functions and arrays.

Some changes I initially attempted to create but decided against:

* I created another random number generator that would be used to determine the user’s choice but both this generator and the one that would determine the computer’s choice continued to receive and output he same answer. I decided that it would be better for the user to at-least choose their own option because without their choice it kind of lost the feeling of a game.
* An array that would contain the number of games played in a row then output the total number in a file for the user to see, decided against it because I believed it was too simple a use of an array.

Unlike the first project I encountered various errors throughout attempting this project. The majority of my difficulty came from the implementation of functions. My first attempt at a function was a simple input validation. I was going to move the input validation concerning the player’s choice in the game outside of main to create a fairly simple function for starters, however it would not compile. I then decided to attempt to separate the various if/else if statements that determined the winner of the game into various functions however once again it would not compile. after many moments of frustration I realized that my error was that when I attempted to call upon a function instead of calling lizard(); I was calling void lizard();I then once again attempted to produce functions that would output the various block of if statements that determined the victory of the game only for the compiler to tell that there weren’t enough arguments in my function to continue. For the sake of time I then proceeded to make 5 simple functions, rock, paper, scissors, lizard, spock. Each function when called, displayed a graphic similar to its name. I decided that if I still had time I would go back and attempt the more complex functions that were expected of us.

I could not figure out how to use arrays in the context of the RPSLS game so I decided to make a second simple game tic tac toe, which would focus on the implementation of arrays. The computer would be o and the user would be x. The plan was to produce and output an array with the size of 9 integers. It would be formatted into the shape of a 3X3 square. The user would then choose a spot square, and then using a random number generator the computer would choose a spot on the square. This would continue until someone got three in a row and won. Then they would be offered the choice to play again.

Pseudo Code:

*Get basic information from the user in order to develop some sort of username*

*Output that username in a file*

*Provide the user with the two game options (TTT, RPSLS)*

*TTT*

*User is playing as x computer is playing as o*

*Seed random number generator that will continue to provide the computer’s choice through out the game.*

*Create array, fill with single integer.*

*Ask the user to input where they would like to place their x*

*Place their x*

*Randomly place the computer’s o*

*Make sure their choices cannot over lap*

*Continue until one gets three in a row (horizontal, vertical, diagonal)*

*Promt user to play again or quit*

*RPSLS*

*Provide the user with their choices*

*Get their choice and inform them of what they have chosen*

*Inform them of what the computer has randomly selected*

*Compare the choices to determine who won*

*Output results*

*Output graphics*

*Promt user to play again or quit*

Flowchart:

