Taurean Roberts

October 25, 2017

Assessment One

**Tic-Tac-Toe Documentation**

//Prototype: char Grid2D[3][3] = {{'7','8','9'},{'4','5','6'}, {'1','2','3'}}  
 //Description: Hard sets the values to match the number pad on the keyboard  
 //Arguments: N/A  
 //Precondition:Hard Sets values in the grid  
 //Postcondition: Gives user some visual reference to the positions  
 //Protection Level: Private  
  
 //Prototype: char mPlayer;  
 //Description: initialize the Player turn rules  
 //Arguments: N/A  
 //Precondition: Covers the player cycle process.  
 //Postcondition: Returns the player piece rotation  
 //Protection Level: Private  
  
 //Prototype: void DisplayGrid();  
 //Description: Shows the area in which the grid will appear  
 //Arguments: N/A  
 //Precondition: Gives the values that are being placed boundries to not pass.  
 //Postcondition: Allows user input from the PlayPiece() function personal slot  
 //Protection Level: Public  
  
 //Prototype: bool PlayPiece();  
 //Description: Functions as a rotator for two players in the game.  
 //Arguments: N/A  
 //Precondition: Starts of with player one with the "X" piece and second with "O"  
 //Postcondition: Covers the numbers with a "X" or a "O" to show placement.  
 //Protection Level:Public  
  
 //Prototype: void DrawGrid();  
 //Description: Updates the grid with the pieces that were placed in the slots.  
 //Arguments: N/A  
 //Precondition: Sees the input and checks if its been placed in the grid  
 //Postcondition: Creates a visual history of player pieces for the current match  
 //Protection Level: Public  
  
 //Prototype: bool WinCondition();  
 //Description: Checks Win Condition  
 //Arguments: N/A  
 //Precondition: Runs game until someone meets the win condition  
 //Postcondition: Ends application  
 //Protection Level: Public

**String Class Documentation**

//Prototype: const char \*mString;  
 //Description: Custom String class structure  
 //Arguements: N/A  
 //PreCondition: Used to give the string a place.  
 //PostCondition: Connects the functions together.  
 //Protection Level: Private  
  
 //Prototype: MyString MyString.  
 //Description: A String with a default mString is created.  
 //Arguments: N/A  
 //Precondition : N/A  
 //Postcondition : A default String is created.  
 //Protection Level: Public  
  
 //Prototype: Mystring Mystring(char \*string)  
 //Description: A MyString constructor with a arguement pointing to the constant char.  
 //Arguements: Takes in a char with a pointer to mString.  
 //PreCondition: N/A  
 //PostCondition: A string is being assigned to a varible.  
 //Protection Level: Public  
  
 //Prototype: int GetLength();  
 //Description: Returns the numbers of characters stored in the string  
 //Arguements: N/A  
 //Preconditions: Makes a new string long until it hits the end term character. '\0'  
 //PostConditions: Gives a pre determined length base on the size of the string.  
 //Protection Level: Public  
  
  
 //Prototype: MyString ToUpper();  
 //Description: Gives the string some uppercase letters  
 //Arguements: N/A  
 //PreCondition: A string with non unified casing.  
 //PostCondition: All the casing in the string is uppercased  
 //Protection Level: Public  
  
 //Prototype: MyString ToLower();  
 //Description:Gives the string some lowercase letters  
 //Arguements: N/A  
 //PreCondition: A string with non unified casing.  
 //Postcondition: All the casing in the string is lowercased.   
 //Protection Level: Public  
  
 //Prototype: bool CompareStrings(MyString &other);  
 //Description: Compares the strings together  
 //Arguements: N/A  
 //PreCondition: Takes two strings and checks to see if they are the same length and have the same characters.  
 //PostCondition: Returns true when the two strings are the same length and characters.  
 //Protection Level: Public  
  
 bool FindSubString(MyString subString, int index = 0);  
 //Prototype: bool FindSubString(MyString subString, int index = 0);  
 //Description: Checks to see if the string has part of the info requested. ex: "min" is in "Gaming" true.  
 //Arguements: A subString of type MyString and a index of the type int that is set on default 0  
 //PreCondition: Takes a second string and compares to see if each character is the same in that order.  
 //PostCondition: Returns true if the substring matches. If not it breaks.  
 //Protection Level: Public  
  
 MyString ReplaceSubString();  
 //Prototype: MyString ReplaceSubString();  
 //Description: Takes part of the string and replaces that part with the new part- Bonus  
 //Arguements: N/A  
 //PreCondition: N/A  
 //PostCondition: N/A  
 //Protection Level: Public  
  
 MyString AppendStrings(MyString aString);  
 //Prototype: MyString AppendStrings(MyString aString);  
 //Description: Takes two strings and places the first one together next to the second.  
 //Arguement: Takes in a MyString called aString.  
 //PreCondition: Takes the first string and puts it behind the second string.  
 //PostCondition: Merges the two strings in the opposite order  
 //Protection Level: Public  
   
 MyString PrependStrings(MyString aString);  
 //Prototype: MyString PrependStrings(MyString aString);  
 //Description: Takes two strings and places the second string before the first string.  
 //Arguements: Takes in a MyString called aString.  
 //PreCondition: Takes the second string and puts it in front the other  
 //PostCondition: Merges the two strings in the opposite order  
 //Protection Level: Public  
  
 char GetIndex(int place);  
 //Prototype: char Getindex(int place);  
 //Description: Finds the strings position in the argument passed in.  
 //Arguments: An int taking the place of origin in the string  
 //Precondition: A string.  
 //Postcondition: The character at the point in the string is returned.  
 //Protection Level: Public  
  
 friend istream &operator >> (istream &input, MyString &other);  
 //Prototype: friend istream &operator >> (istream &input, MyString &other);  
 //Description: Gets user input and allows them to put it in the string  
 //Arguements: What the string is getting changed too.  
 //Precondition: User inputs a string to be tested  
 //PostCondition: Program will run smoothly if it passes all the tests  
 //Protection Level: Public