Welcome to Learn C# Week 3 Working with Start Screens

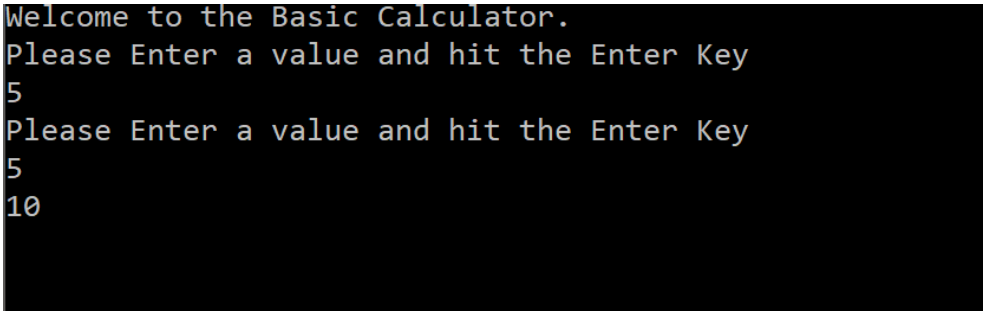
This week we will work on setting up menus. Whether you've noticed or not, Start Screens are an important part of video games and programs. This week we will implement a Start Screen similar to that of a game like Pokémon or even Skyrim but first we need to go over last weeks homework. Below is the solution to the task assigned.

```
static void Main(string[] args)
{
    //Create three variables of type int and assign the value of 0 to them
    int numb1 = 0;
    int numb2 = 0;
    int numb3 = 0;

    //Title Message
    Console.WriteLine("Welcome to the Basic Calculator.");

    //Get input and put it in the integer variable numb1
    Console.WriteLine("Please Enter a value and hit the Enter Key");
    numb1 = Int32.Parse(Console.ReadLine());

    //Get input and put it in the integer variable numb2
    Console.WriteLine("Please Enter a value and hit the Enter Key");
    numb2 = Int32.Parse(Console.ReadLine());

    //Assign the value of numb2 + numb1 to numb3
    numb3 = numb2 + numb1;

    Console.WriteLine(numb3);

    //Pauses the program or it will exit immediatly.
    Console.ReadLine();
}
```

First of all, I have to apologize to you guys. The Readline command doesn't work for integers unless something extra is added. Like we learned last week C# requires **data types** for every **variable**. ReadLine(); does not work for integer values, only for strings. For this reason we will have to look elsewhere in order to get integers as input. Thus the solution to some of your issues is the extension from **Console.ReadLine();** to **Int32.Parse(Console.Read());** At the end of last weeks lesson I did say that you could ask for help or look for a solution online and since there are no complaints in the comments am going to assume you guys found the solution to the problem online.

That being said there may be others who are upset and are saying "How was I supposed to know that?" the answer is you shouldn't/couldn't and you should get used to it. Professionals in the industry don't remember every command by heart, in fact they often look up the how to write some commands and what they do. As an example there has even been several occasions when my professors in a university setting took 10 minutes to search online the issues to their code in a room full of 100+ students. We can't remember

everything and don't expect you to memorize everything as well. Just try to practice often and learn to search for solutions online. A popular site people in the industry explore is Stack Overflow. Made to search for your issues, Stack Overflow is full of people who will shoot down any problem you may have if it's already up on the site.



Here is my solution's result on the console. If you enter 5 twice and your result is 10 then your program works.

Your solution doesn't have to look anything like mine, there are millions of ways to write code and not everyone will have the exact same code. As long as the output is similar you can count yourself as someone who passed the assignment. Now I want to also mention a possible logic error you guys may have run into. Some of you may have overlooked my lesson and just used string variables. That's some of you may have entered **string numb1 = 0;** or **string numb1;** For string numb1 = 0; you would have gotten an issue from the Visual Studio telling you can't set integers to strings. As stated in the last lesson, strings are only for words not numbers. For string numb1; you could ignore the issues Visual Studio would have given you and it all would have worked, or would it?

```
//Create three variables of type string.
string numb1;
string numb2;
string numb3;

//Title Message
Console.WriteLine("Welcome to the Basic Calculator.");

//Get input and put it in the integer variable numb1
Console.WriteLine("Please Enter a value and hit the Enter Key");
numb1 = Console.ReadLine();

//Get input and put it in the integer variable numb2
Console.WriteLine("Please Enter a value and hit the Enter Key");
numb2 = Console.ReadLine();

//Assign the value of numb2 + numb1 to numb3
numb3 = numb2 + numb1;

Console.WriteLine(numb3);

//Pauses the program or it will exit immediatly.
Console.ReadLine();
```

Take a look at the code above. It's nearly identical to the one before but I have the variables at the top with the data types set as strings and just use regular old Console.ReadLine(); Visual Studio won't tell you anything is wrong. There is no issues with the syntax, this is just a logical error on the programmers part. Some of you are thinking well if there are no issues showing up with red squiggly lines then what's the problem? Well adding two integers together is a lot more different than adding to strings together. Let's take a look at the output of this program if we enter 5 for both variables again. Will we get 10 again?

```
Welcome to the Basic Calculator.
Please Enter a value and hit the Enter Key
5
Please Enter a value and hit the Enter Key
5
55
```

As you can see with the image above we don't get the same results. 5 + 5 does not equal 55 at least when it comes to numerical values. String addition is different and something that is referred to in programming as concatenation. It's a fancy word for combining words together. Play with the program and you'll see the first value will be mashed together with the next value. This is a good way to get the name of a player at the start of a game. Enter

your first name and then last name and the program will display both your first name and last in one line.

Moving on to the actual lesson, this week we are going to build on what we did before which was get the beginning data of a new player by asking for the name and gender. I know it sounds out of order but games and films are made in order so lets begin building the Start Screen.

So let's start again this time but now we will dive into conditional branches and loops. The first loop we will learn about is the **While loop**. This loop does exactly what the name implies. We will set a condition and check if it's true or false. The loop will go around infinitely until the condition is false. In our case we will use **!= not equal**.

While(begin != "Start"){

}

The loop works by putting in code you want to repeat inside the curly braces { }. Once it reaches the end curly brace it will go back to the beginning line, check the condition, if the condition is true it will repeat the loop. Write down the code snippet below in a new project.

```
//Create a variable of type string named begin and gave it a default value.
string begin = "default";

//Title Message of Our Game
Console.WriteLine("Pokemon");

//Here we wrote our while loop. In the parenthesis it checks if the statement is true or not
//Since it is true that begin does not contain the string Start in it, it will loop on
//forever until it is false.
while (begin != "Start") {

    //Give a line of space
    Console.WriteLine();

    //Prompt the user to  to type begin
    Console.WriteLine("Type Start to Begin");

    //Set the variable begin to to the input the user enters.
    begin = Console.ReadLine();
}

//Pause the program or it will shut down immediatly.
Console.ReadLine();
```

Once you've written the program exactly as is try running the program by hitting the debug green start arrow button in Visual Studio.

This program emulates the start screen of a basic Pokémon game, or any other game with a main menu for that matter. We began by initializing a variable of type string and gave it a value of the word default. We then wrote the title of our game which you can replace if you want and continued on to write our While loop. The while loop checks if the condition **begin != "Start"** is true. Because we set the default value of begin equal to "default" the condition is true that begin does not equal to start and so the loop begins. Inside the loop the code inside is executed. Notice how we indented the code inside of the loop. This is for clarity, other people will one day see you code and indentation helps people read and keep track of where loops begin and end.

Also since we are on topic of clarity, notice the comments I added to my program. All the lines in green are text I wrote that the compiler ignores but people like you can read and understand. It's a good way to help others a heads up of what is going but as a warning I should tell you in the real world it's frowned upon to write as much as I did. Your code should be self explanatory and neat and for cases where someone may indeed get lost that's when you write a comment to remind others or yourself what's happening in the code. Because we are learning C# from the ground up feel free to write as much as you want as long as it helps you understand what's happening.

Back in the While loop we continue on to write extra code that will hopefully change the condition so that it reads false on the next turn. We prompt the user to type Start and then change the value in begin to whatever the user enters. If the user typed Start it will exit the loop because **begin != "Start"** will then be false. In regular English we would say begin does not equal to start but that is no longer true because we the user typed Start. Thus the while loop will skip the code block within and skip the end curly brace. If the user types anything but Start they will be inside that loop until they give the correct string.

Now that we know how while loops work lets expand on it just a little bit. Notice how if we run the code and try to type start when it asks us to type Start it won't work and the loop will go on forever? The reason for this is because capital and lower case letters are not the same inside a computer, the letter S and s are completely different in the eyes of a machine. Lets fix this small issue by adding another condition to the **While** loop. In last weeks lesson we took a glance at the **== equals**condition today we used the new condition **!= does not equal** now how about we use the **&& AND** condition.

while(begin != "Start" && begin != "start"){

}

This time we modified the conditions the while loop checks so that it looks if the variable begin does not hold the string Start and does not hold the string start. The while loop will check both conditions and make sure if Start AND start or not equal to whatever is inside the begin variable to loop on forever. If either Start or start is in the variable then it's false that begin != Start or start making the loop exit.

```
static void Main(string[] args)
{
    //Create a variable of type string named begin and gave it a default value.
    string begin = "default";

    //Title Message of Our Game
    Console.WriteLine("Pokemon");

    //Here we wrote our while loop. In the parenthesis it checks if the statement is true or not
    //Since it is true that begin does not contain the string Start in it, it will loop on
    //forever until it is false.
    while (begin != "Start" && begin != "start") {

        //Give a line of space
        Console.WriteLine();

        //Prompt the user to  to type begin
        Console.WriteLine("Type Start to Begin");

        //Set the variable begin to to the input the user enters.
        begin = Console.ReadLine();
    }

    //Pause the program or it will shut down immediatly.
    Console.ReadLine();

}
```

Hopefully by now you start to see how powerful these conditional branches, loops, and conditions are in programming. These are the essential building blocks to programmers and if you can get this concept down you pretty much have 30-40% of programming down.

Now that you got that down lets do a new homework assignment that builds off last weeks homework. You already have the solution for last weeks assignment above, now try putting

a while loop in your calculator that will continue to add values together until a user types in exit.

Specifications:

1. Display the Title of your program ex. “Welcome to the Calculator Program”
2. Prompt the User to enter a value for a integer variable.
3. Assign the value into a variable.
4. Prompt the user to enter another value for another integer variable.
5. Assign the value into another variable.
6. Add those variables together and put them in a third variable.
7. Display the contents of that third variable
8. **NEW Ask the user if they want to exit to type “exit”.**
9. If the user enters anything other than exit they while loop should sent them back to step 2.
10. If the user enters exit then the while loop should end and you should finish the program by displaying the message “Goodbye”

Overall nothing new is in this assignment. You just need to know where to place the While loop, set its conditions, and get the condition data inside the loop so that you can exit the loop.

Again you can look online for help, ask questions here, ask me on Twitter, or just wait until next week when I post the solution. Next weeks lesson will be linked here when it is posted.