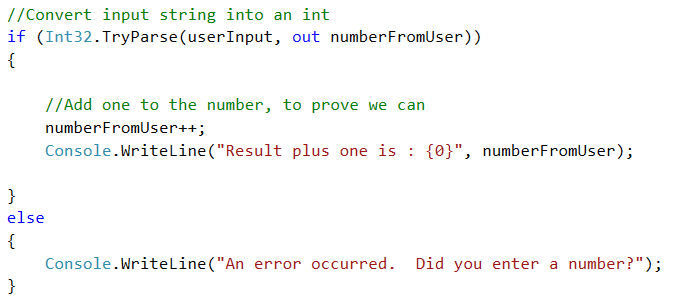
**Bad Data**

|  |  |  |  |
| --- | --- | --- | --- |
| Line: | Input Line | [Easy Difficulty](http://fstcat-csharp-exercises.azurewebsites.net/Exercises/IconGlossary.html#DIFF) [Input](http://fstcat-csharp-exercises.azurewebsites.net/Exercises/IconGlossary.html#IN) [Learning Outcome One](http://fstcat-csharp-exercises.azurewebsites.net/Exercises/IconGlossary.html#LO) [Walkthrough](http://fstcat-csharp-exercises.azurewebsites.net/Exercises/IconGlossary.html#WT) | |
| Type: | Walkthrough |
| You should have completed: | [**Temperature Conversion**](http://fstcat-csharp-exercises.azurewebsites.net/Exercises/TemperatureConversion.html) | This topic leads to: | [**Verification**](http://fstcat-csharp-exercises.azurewebsites.net/Exercises/Verification.html) |

**Summary**

So far, if a user has typed in information in a format the program was not expecting, it has crashed our programs. There is a robust approach to error handling within C# but we will cover it later in the module. In the meantime, there is an approach which will help reduce the number of crashes due to incorrect user input.

**Task**

1. Open your program from the Getting Numbers exercise.
2. Instead of using Int32.Parse to convert the user input to a number, use the following.  
   

**Questions**

1. Why is there an 'out' keyword there?
2. How would you use the same technique to get a float or double value from the user?