**Topic 3: Input Validation**

Top of Form

Bottom of Form

**Content**

**Reading**

[Reading](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286910_1)

String Conversions

* + Microsoft:docs.microsoft.com
    - [Convert strings to numbers](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/how-to-convert-a-string-to-a-number)
      * [TryParse Int](https://docs.microsoft.com/en-us/dotnet/api/system.int32.tryparse?view=netcore-3.1)
      * [TryParse Double](https://docs.microsoft.com/en-us/dotnet/api/system.double.tryparse?view=netcore-3.1)
  + GeeksForGeeks: geeksforgeeks.com
    - [Different ways to convert string to numbers](https://www.geeksforgeeks.org/different-ways-to-convert-string-to-integer-in-c-sharp/)
  + TutorialsPoint: tutorialspoint.com
    - [C# int.TryParse](https://www.tutorialspoint.com/chash-int-tryparse-method)
  + W3Schools: w3schools.com
    - [C# Type Casting](https://www.w3schools.com/cs/cs_type_casting.asp)

**TryParse**

[TryParse](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286910_1)

TryParse methods are used to validate the type of data entered by the user. This does not mean the user input is accurate or an acceptable value. A phrase you may hear is “garbage in, garbage out.” (GIGO) Since the computer cannot tell if the input is good, it is the programmer's responsibility.

https://youtu.be/zFTHmLp0k24

using System;

namespace InputValidation

{

class Program

{

private static double TestPercentage(int score)

{

return score / 100.0;

}

static void Main(string[] args)

{

string testScoreString;

int testScore;

// prompt user for input

Console.WriteLine("Please enter the test score");

testScoreString = Console.ReadLine();

if (int.TryParse(testScoreString, out testScore))

{

// Continue to process the input.

Console.WriteLine("Test Percentage: {0:P}", TestPercentage(testScore));

}

else

{

Console.WriteLine("The test score must be an integer.");

}

}

}

}

For instance an age of -3 for a human would not be good input, but it is an acceptable integer. This program would accept -3, so more input validation is needed.

**Business Logic Input Validation**

[Business Logic Input Validation](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286910_1)

TryParse helped with validating the data type. It did not ensure the at the user input follows business logic or common logic. For instance, if you ask for an age in years, and the data type is integer, users can still input negative values. Even if your programs prompts the user for the cost of an item, validating with TryParse that it is a double or decimal does not validate a reason price (not negative, not a billion dollars).

You can use versions of the if statement to further validate the input. For example, asking for an test score 0-100.

https://youtu.be/vsKIuz-ep9U

using System;

namespace InputValidation

{

class Program

{

private static double TestPercentage(int score)

{

return score / 100.0;

}

static void Main(string[] args)

{

string testScoreString;

int testScore;

// prompt user for input

Console.WriteLine("Please enter the test score (0-100)");

testScoreString = Console.ReadLine();

if (int.TryParse(testScoreString, out testScore))

{

if (testScore >= 0 && testScore <= 100)

{

// Continue to process the input.

Console.WriteLine("Test Percentage: {0:P}", TestPercentage(testScore));

}

else

{

Console.WriteLine("Test score must be in the range 0 − 100.");

}

}

else

{

Console.WriteLine("The test score must be an integer.");

}

}

}

}

Right now, your exits when an error is encountered or sets a default value for the failed user input gathering. In the next module you will learn loops to be able to continue to prompt the user for valid input until an acceptable input is supplied by the user.

**Input Validation Practice**

[Input Validation Practice](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286910_1)

It is important to validate user input to make your program robust. Instead of exiting with a confusing (to the user) error message, you should print an error message explaining the problem. It's often best to continue the program when an input error is encountered.

Write a Console App that builds a character sheet for gaming. Since we don't have GUI building in our toolbox yet, you can print out a paragraph of the character.

For character building, individuals roll dice to r to set values. When you run the code, you can select random numbers in the appropriate value range to test your code. Don't forget to also select improper input to test your input validation.

The code will first prompt the user for their character name. Next, it will ask the user for their character information from the assumed dice roll. (Later when you learn about random numbers and loops, you can remove this part of user input and generate the character numbers with simulating the roll of the dice.)

  Business Logic: Include the following characteristics: charm, strength and shield. Charm is a number in the range 10 to 20. Strength is between 0 and 40. Shield is in the range 1 to 10.

  Your program must use TryParse and compound conditionals to validate the input. If the range is incorrect, select a valid value for the user (lowest, median, or any number).

See attached character sheet as a reference if you have not seen any [Dungeons and Dragons](https://g.co/kgs/9SgzCk)  type games for Wendy's Free D&D game. [Wendys\_Feast\_Of\_Legends\_Character\_Sheets.pdf](https://dmacc.blackboard.com/bbcswebdav/pid-7286930-dt-content-rid-101466151_1/xid-101466151_1) [Wendys\_Feast\_Of\_Legends\_Character\_Sheets.pdf - Alternative Formats](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286910_1)

Diagram

Description automatically generated

**Sample Output for Character Building with Input Validation**

[Sample Output for Character Building with Input Validation](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286910_1)

Sample output (You do not need to be so verbose when asking for name, strength, charm, and shield):

https://youtu.be/6KGuwwVytRY

Good Input:

Text

Description automatically generated

Invalid input, non-numeric:

Text

Description automatically generated

Numeric input, invalid range:

Text

Description automatically generated