Number to Roman Numeral Converter in C#

Introduction

Master number conversion with C# and the power of the switch statement! This lesson reveals a clean and efficient approach to building your Number to Roman Numeral Converter. We will focus on converting a given integer (ranging from 1 to 10) to its Roman numeral equivalent using a switch statement for each digit and combining the results. By leveraging the switch statement, we can simplify the conversion process and make our code more readable and maintainable.

The switch statement in C# provides a concise and structured way to handle multiple cases based on the value of an expression. In the context of our Number to Roman Numeral Converter, we can use the switch statement to handle each digit of the input number individually and map it to its corresponding Roman numeral representation. This approach allows us to break down the conversion process into smaller, manageable steps, making our code easier to understand and modify.

Let's dive into the details of how to implement this Number to Roman Numeral Converter using a switch statement in C#. By the end of this lesson, you will have a solid understanding of the conversion logic and be able to create your own converter with confidence. So, let's get started and unlock the power of the switch statement in C#!

Objectives

Our focus will be on building a Number to Roman Numeral Converter that can accurately convert a given integer (1-10) to its Roman numeral equivalent. Through this lesson, we aim to provide you with a comprehensive understanding of the conversion process while emphasizing the advantages of utilizing the switch statement for handling each digit. By the end of this lesson, you will not only understand the logic behind the conversion but also gain hands-on experience by practicing and applying your knowledge in creating your own Number to Roman Numeral Converter.

Objectives:

1. Understand the logic behind converting a given integer to its Roman numeral equivalent: We will delve into the principles and rules governing the conversion process, enabling you to comprehend the step-by-step logic required to convert a number into its Roman numeral representation. By understanding the underlying principles, you will be equipped with the knowledge necessary to tackle more complex conversion scenarios in the future.
2. Learn the implementation of a Number to Roman Numeral Converter using a switch statement: You will learn how to utilize the switch statement in C# to handle each digit of the input number individually and map it to its corresponding Roman numeral representation. Through detailed

explanations and examples, you will gain a solid understanding of how to effectively leverage the switch statement for efficient and readable code.

3. Practice and apply your knowledge: To reinforce your understanding, you will have the opportunity to practice implementing the Number to Roman Numeral Converter on your own. By working through coding exercises and challenges, you will strengthen your skills in using the switch statement and gain confidence in your ability to convert numbers to Roman numerals.

4. Apply your knowledge to real-world scenarios: Building a Number to Roman Numeral Converter is not only a valuable exercise in understanding number conversion but also a practical skill with various applications. You will learn how to incorporate the converter into your own projects, such as creating user interfaces or handling historical data. By applying your knowledge in real-world scenarios, you will gain a deeper appreciation for the power and versatility of the switch statement in C#.

Through a combination of comprehensive explanations, hands-on practice, and real-world applications, this lesson will empower you to become proficient in converting numbers to Roman numerals using the switch statement in C#.

Source code example

```
1.  using System;
2.
3.  namespace RomanNumeralConverter
4.  {
5.      class Program
6.      {
7.          static void Main()
8.          {
9.              Console.WriteLine("iNetTutor.com -
     Number to Roman Numeral Converter");
10.
11.             Console.Write("Enter a number (1-10): ");
12.             int number = int.Parse(Console.ReadLine());
13.
14.             // Validate input
15.             if (number < 1 || number > 10)
16.             {
17.                 Console.WriteLine("Invalid input. Please enter a number between
    1 and 10.");
18.             }
19.             else
20.             {
21.                 string romanNumeral = "";
22.
23.                 // Switch statement for each digit
24.                 switch (number)
25.                 {
26.                     case 1:
27.                         romanNumeral += "I";
```

```
                            break;
                        case 2:
                            romanNumeral += "II";
                            break;
                        case 3:
                            romanNumeral += "III";
                            break;
                        case 4:
                            romanNumeral += "IV";
                            break;
                        case 5:
                            romanNumeral += "V";
                            break;
                        case 6:
                            romanNumeral += "VI";
                            break;
                        case 7:
                            romanNumeral += "VII";
                            break;
                        case 8:
                            romanNumeral += "VIII";
                            break;
                        case 9:
                            romanNumeral += "IX";
                            break;
                        case 10:
                            romanNumeral += "X";
                            break;
                    }

                    Console.WriteLine($"Roman Numeral: {romanNumeral}");
                }

                Console.ReadLine();
            }
        }
}
```

```
01. using System;
02.
03. namespace RomanNumeralConverter
04. {
05.     class Program
06.     {
07.         static void Main()
08.         {
09.             Console.WriteLine("iNetTutor.com - Number to Roman Numeral Converter");
10.
11.             Console.Write("Enter a number (1-10): ");
12.             int number = int.Parse(Console.ReadLine());
13.
14.             // Validate input
15.             if (number < 1 || number > 10)
16.             {
17.                 Console.WriteLine("Invalid input. Please enter a number between 1 and 10.");
18.             }
19.             else
20.             {
21.                 string romanNumeral = "";
22.
23.                 // Switch statement for each digit
24.                 switch (number)
25.                 {
26.                     case 1:
27.                         romanNumeral += "I";
28.                         break;
29.                     case 2:
30.                         romanNumeral += "II";
31.                         break;
32.                     case 3:
33.                         romanNumeral += "III";
34.                         break;
35.                     case 4:
36.                         romanNumeral += "IV";
37.                         break;
38.                     case 5:
39.                         romanNumeral += "V";
40.                         break;
41.                     case 6:
42.                         romanNumeral += "VI";
43.                         break;
44.                     case 7:
45.                         romanNumeral += "VII";
46.                         break;
47.                     case 8:
48.                         romanNumeral += "VIII";
49.                         break;
50.                     case 9:
51.                         romanNumeral += "IX";
52.                         break;
53.                     case 10:
54.                         romanNumeral += "X";
55.                         break;
56.                 }
57.
58.                 Console.WriteLine($"Roman Numeral: {romanNumeral}");
59.             }
60.
61.             Console.ReadLine();
62.         }
63.     }
64. }
```

Explanation

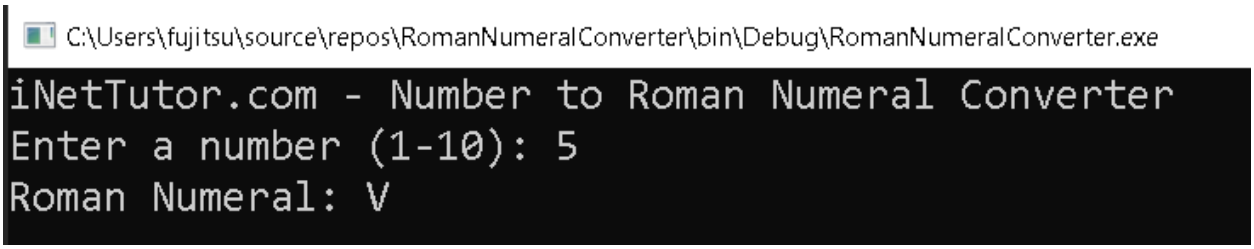
This C# code converts a given integer between 1 and 10 to its corresponding Roman numeral equivalent. Here's a breakdown of the code:

1. Introduction:
   - using System;: This line includes the standard .NET libraries.
   - namespace RomanNumeralConverter: This line defines the namespace for the code.
   - class Program: This line defines the main class of the program.
   - static void Main(): This is the entry point of the program, where the code execution starts.
2. Welcome Message:
   - Console.WriteLine("iNetTutor.com - Number to Roman Numeral Converter");: This line prints a welcome message with the website name.
3. User Input and Validation:
   - Console.Write("Enter a number (1-10): ");: This line prompts the user to enter a number.
   - int number = int.Parse(Console.ReadLine());: This line reads the user's input, converts it to an integer, and stores it in the number variable.
   - if (number < 1 || number > 10): This condition checks if the input is between 1 and 10.
     - If invalid, it prints an error message and stops the program.
     - If valid, it proceeds to convert the number.
4. Roman Numeral Conversion:
   - string romanNumeral = "";: This line initializes an empty string to store the converted Roman numeral.
   - switch (number): This switch statement efficiently handles different single-digit numbers (1-10).
     - Each case corresponds to a digit and appends the corresponding Roman numeral string to the romanNumeral.
     - The break statement ensures the code exits the switch after each case execution.
5. Output:
   - Console.WriteLine($"Roman Numeral: {romanNumeral}");: This line prints the final Roman numeral equivalent stored in romanNumeral.
6. Closing:
   - Console.ReadLine();: This line pauses the program before closing, allowing the user to see the output.

Additional Notes:

- This code only handles numbers between 1 and 10. You can extend it to handle larger numbers with more switch cases or different logic.
- The code could be further improved by adding more informative error messages for different types of invalid input.

Output



Summary

This C# code transforms single-digit integers (1-10) into their corresponding Roman numeral equivalents. It demonstrates the power of the switch statement for clear and efficient conversion logic.

The code starts by introducing itself and then prompts the user for a number between 1 and 10. It validates the input, ensuring it falls within the acceptable range. If valid, a switch statement takes over, acting as a decision-making hub. Each case within the switch corresponds to a specific digit (1-10), and its corresponding Roman numeral string is appended to an empty string variable. This step-by-step approach using cases ensures accurate conversion for each digit. Finally, the assembled Roman numeral is displayed, revealing the ancient counterpart of the entered number.

This code offers several advantages:

- Clarity: The switch statement provides a clear and readable way to map digits to their Roman numeral representations.
- Efficiency: By using separate cases for each digit, the code avoids complex conditional logic, which can improve performance and maintainability.
- Simplicity: The overall structure is straightforward and easy to understand, making it suitable for both beginners and experienced programmers.

While this code focuses on single-digit numbers, it serves as a strong foundation for further exploration. You could expand it to handle larger numbers by adding more cases or implementing alternative logic. Additionally, you could explore different Roman numeral formatting styles or enhance error handling for various input scenarios.

By understanding and potentially modifying this code, you gain valuable experience with both Roman numeral conversion and the benefits of the switch statement in C#.

Exercises and Assessment

Exercise 1: Code Refactoring One way to improve the code is by refactoring it to make it more modular and reusable. Currently, the conversion logic is implemented within the Main method. Refactor the code by creating a separate method called ConvertToRomanNumeral that takes an integer as input and returns the Roman numeral equivalent as a string. Update the Main method to call this new method and display the result.

Exercise 2: Error Handling Enhance the code by adding additional error handling for invalid user input. Currently, the code checks if the number is within the range of 1 to 10. Modify the code to handle cases where the user enters non-numeric input or numbers outside the specified range. Display appropriate error messages for these scenarios and prompt the user to enter a valid input.

Exercise 3: Expand the Range Extend the code to handle a wider range of numbers, beyond 1 to 10. Update the switch statement to include cases for numbers up to 1000. Implement the logic to convert these larger numbers to their Roman numeral equivalents using the switch statement.

Assessment: To assess the improvement of the code, consider the following criteria:

1. Modularity: Has the code been refactored to separate the conversion logic into a separate method?
2. Reusability: Can the code be easily reused in other parts of the program or in different projects?
3. Error Handling: Does the code handle invalid user input effectively and provide appropriate error messages?
4. Range Expansion: Has the code been extended to handle a wider range of numbers beyond the initial range of 1 to 10?

By evaluating the code against these criteria, you can determine the effectiveness of the improvements made and identify areas for further enhancement.

Meta Description

Master the Number to Roman Numeral conversion in C# using a switch statement. Understand, apply, and practice with this comprehensive guide and code example.