Month Display Program in CSharp

Blog post introduction

Welcome, C# enthusiasts! In this blog post, we will explore a simple yet practical C# console program that prompts users to input a number representing a month and displays the corresponding month name. This exercise is a great opportunity to reinforce your understanding of basic control flow structures in C#. As both an instructor and content creator, my goal is to provide you with a clear and detailed explanation of the code, helping you enhance your programming skills.

Introduction

The core idea of this program is to prompt the user to input a number from 1 to 12, representing a month. The program will then determine the corresponding month name and display it. For instance, if the user inputs '3', the program will output 'March.' Additionally, the program is equipped to handle invalid inputs gracefully, providing feedback and prompting the user to re-enter a valid number.

If-Else If Statement

At the heart of this program lies the if-else if-else statement, a powerful construct in C# that enables decision-making based on different conditions. In the context of our program, each 'if' or 'else if' block checks whether the user input matches a specific number corresponding to a month. If a match is found, the associated month name is displayed. The 'else' block acts as a catch-all, handling cases where the input doesn't match any valid month number, providing a response for invalid input.

In essence, the if-else if-else statement allows the program to navigate through different possibilities based on the user's input, ensuring a meaningful and accurate output. It's a fundamental control flow structure that enhances the program's flexibility and responsiveness.

As we delve deeper into the source code, you'll gain a comprehensive understanding of how this if-else if-else structure is implemented to create a user-friendly and robust month display program in C#. Let's embark on this learning adventure together!

Objectives

The primary objectives of this lesson are designed to empower you with essential C# programming skills, fostering a comprehensive understanding of user interaction, decision-making, and error handling. By the end of this lesson, you will be able to:

1. Efficiently Capture User Input:

    - Learn the techniques to prompt users for input in a C# console application.
    - Understand how to utilize the Console.ReadLine() method to acquire user data.

- Explore the role of data types in input validation and user communication.

2. Master Decision-Making with if-else if-else Statements:
    - Grasp the concept of conditional statements and their pivotal role in program flow control.
    - Implement if-else if-else statements to create a branching mechanism that directs the program's execution based on specific conditions.
    - Develop an intuition for selecting the appropriate conditional structure to cater to different scenarios.
3. Dynamic Display of Month Names:
    - Gain hands-on experience in associating numeric input with corresponding month names.
    - Understand the role of variables in storing and managing data within a program.
    - Enhance your ability to design logic that transforms user input into meaningful output.
4. Graceful Handling of Invalid Input:
    - Acquire skills in input validation to ensure the program receives expected data types and values.
    - Implement loops for continuous user input until valid data is provided.
    - Enhance the user experience by providing clear and helpful feedback when input falls outside the expected range or format.
5. Comprehensive Error Feedback:
    - Develop strategies to communicate errors effectively to users.
    - Learn to differentiate between various types of errors, such as non-numeric input or values outside the specified range.
    - Explore the significance of user-friendly error messages in improving the overall usability of your programs.

By focusing on these objectives, you'll not only build a foundation in user interaction and decision-making in C# but also develop practices for creating robust and user-friendly applications. This lesson is crafted to ensure a well-rounded understanding, equipping you with skills applicable to a wide range of programming scenarios.

Source code example

```
using System;

namespace MonthDisplayProgram
{
    class Program
    {
        static void Main()
        {
            // Ask the user to enter a number from 1 to 12
            Console.Write("Enter a number from 1 to 12: ");
            int monthNumber;

            // Validate user input
            while (!int.TryParse(Console.ReadLine(), out monthNumber) || monthNumber < 1 || monthNumber > 12)
            {
                Console.Write("Invalid input. Please enter a number from 1 to 12: ");
            }

            // Use if-else if-else statement to determine the month
            if (monthNumber == 1)
            {
                Console.WriteLine("January");
            }
            else if (monthNumber == 2)
            {
                Console.WriteLine("February");
            }
            else if (monthNumber == 3)
            {
                Console.WriteLine("March");
            }
            else if (monthNumber == 4)
            {
                Console.WriteLine("April");
            }
            else if (monthNumber == 5)
            {
                Console.WriteLine("May");
            }
            else if (monthNumber == 6)
            {
                Console.WriteLine("June");
            }
            else if (monthNumber == 7)
            {
                Console.WriteLine("July");
            }
            else if (monthNumber == 8)
            {
                Console.WriteLine("August");
            }
            else if (monthNumber == 9)
            {
                Console.WriteLine("September");
            }
            else if (monthNumber == 10)
            {
                Console.WriteLine("October");
            }
            else if (monthNumber == 11)
            {
                Console.WriteLine("November");
            }
            else if (monthNumber == 12)
            {
                Console.WriteLine("December");
            }
            else
            {
                Console.WriteLine("Invalid month");
            }
            Console.ReadKey();
        }
    }
}
```

Explanation

1. Namespace and Class Declaration:

This code begins by declaring a namespace (MonthDisplayProgram) and a class (Program). Namespaces help organize code, preventing naming conflicts, and classes are the fundamental building blocks of C# programs.

2. Main Method:

The **Main** method is the entry point of the program. It's the method that gets executed when the program starts. The **static** keyword means that the method belongs to the class itself rather than an instance of the class.

3. User Input and Validation:

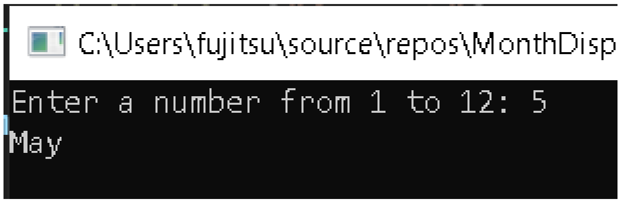
This code prompts the user to enter a number from 1 to 12 and declares a variable **monthNumber** to store the user input.

This **while** loop ensures that the user provides a valid input. It uses **Console.ReadLine()** to read the input as a string and **int.TryParse()** to convert it into an integer. The loop continues until a valid integer within the range of 1 to 12 is provided.

4. If-Else If-Else Statement:

This section uses an if-else if-else statement to determine the month based on the value of **monthNumber**. Each **if** or **else if** block checks whether **monthNumber** is equal to a specific value (representing a month). If a match is found, it prints the corresponding month name. The **else** block handles the case where **monthNumber** doesn't match any valid month, and it prints "Invalid month."

Output



```
Enter a number from 1 to 12: 5
May
```

Summary

In summary, this C# program demonstrates key concepts such as user input, input validation, and conditional statements using if-else if-else. It's a great exercise for understanding basic control flow and user interaction in console applications. Feel free to run the program and experiment with different inputs to observe the outcomes!

Exercises and Assessment

Exercise 1

Objective: Improve input validation

Task: Modify the program to handle non-numeric inputs more gracefully. Implement additional checks to ensure that the user provides a valid numeric input.

Tips:

- Use int.TryParse() to check if the input is a valid integer.
- Provide clear and specific error messages for non-numeric input.

Exercise 2

Objective: Expand month range

Task: Extend the program to handle months beyond December. Allow users to enter a number representing any month of the year (1-12). Update the program to display the corresponding month name accurately.

Tips:

- Adjust the validation and the range of month names accordingly.

Exercise 3

Objective: Explore switch statements

Task: Rewrite the month determination part using a switch statement instead of the if-else if-else structure. Ensure that the functionality remains the same.

Tips:

- Consider the switch statement as an alternative to if-else if-else when dealing with multiple cases.

Quiz

1. **What is the purpose of the int.TryParse method in the provided code?**

   a) To display month names

   b) To validate user input

   c) To perform arithmetic operations

   d) None of the above

2. **How is invalid input handled in the program?**

   a) By displaying an error message

   b) By terminating the program

   c) By prompting the user to re-enter a valid number

   d) Both a and c

3. **In the provided code, what does the while loop primarily do?**

   a) Displays the month names repeatedly

   b) Validates user input and prompts for correction

   c) Performs arithmetic operations

   d) Terminates the program

4. What is the role of the else statement in the program?

a) Display the month name

b) Handle invalid input

c) Prompt the user for input

d) Terminate the program

5. What is the purpose of the Console.Write statement in the program?

a) To display the month names

b) To read user input

c) To perform calculations

d) To handle invalid input