Guess Number Game in CSharp

Guess the Number: Create a simple guessing game where the program generates a random number between 1 and 100, and the user has to guess the number. Provide hints like "Too high" or "Too low" until the user guesses correctly. Use a while loop for this exercise.

Objective: Develop a simple guessing game in C# to reinforce understanding of random number generation, user input handling, and the use of loops.

Introduction

Welcome to an exciting journey into the world of C# programming, where we embark on the creation of a thrilling Guess the Number game. In this lesson, our primary objective is to delve into the fundamental concepts of random number generation, user input handling, and the seamless integration of loops to craft an engaging gaming experience.

Overview of the Lesson's Objective: Our goal is to equip you with the skills needed to develop a dynamic and interactive guessing game, reinforcing your understanding of essential programming constructs. Throughout this exploration, we'll unravel the mysteries of the Random class, unlocking its power to generate unpredictable numbers that lie at the heart of our guessing game.

Introduction to the Guess the Number Game: Imagine a scenario where the computer selects a random number, and your task is to decipher it through a series of informed guesses. This game not only entertains but serves as a practical application of core programming concepts. As we dive into the Guess the Number game, you'll discover the key components that make it tick – from user input mechanisms to the hints provided based on your guesses.

Exploring the Random Class for Generating Random Numbers: Central to our journey is the exploration of the Random class, a powerful tool that enables us to introduce an element of unpredictability into our game. We'll unravel its functionality, understand how it generates random numbers within specified ranges, and leverage this knowledge to infuse excitement and unpredictability into our Guess the Number adventure.

Get ready to flex your programming muscles as we venture into the captivating realm of guessing games, where randomness meets logic, and each guess brings you one step closer to unraveling the mystery number. Let the guessing game commence!

Objectives

This lesson unfolds with a multidimensional objective: to deepen your understanding of key programming concepts, facilitate active learning through hands-on practice, and empower you to apply acquired skills in a real-world gaming scenario.

Objective-Based Education (OBE) Focus:

1. Understand Random Number Generation
   - *Outcome:* Develop a comprehensive understanding of the Random class in C# and its role in generating unpredictable numbers.
   - *Indicator:* Provide detailed explanations on the mechanisms employed by the Random class for number generation.
2. Learn User Input Handling and Validation
   - *Outcome:* Acquire proficiency in handling and validating user input for a guessing game.
   - *Indicator:* Engage in practical exercises that involve user input, emphasizing the importance of validation.
3. Practice Loop Structures in C#
   - *Outcome:* Gain hands-on experience in utilizing while loops for continuous user guessing until the correct number is identified.
   - *Indicator:* Implement and refine loop structures within the Guess the Number game, emphasizing practical applications.
4. Apply Game Development Concepts
   - *Outcome:* Apply learned concepts to create a functional Guess the Number game with dynamic elements such as hints and user feedback.
   - *Indicator:* Demonstrate the ability to translate theoretical knowledge into a tangible and interactive gaming experience.

These four OBE-focused objectives form the cornerstone of this lesson, guiding you to not only understand and learn essential programming principles but also actively practice and apply them in the development of an engaging Guess the Number game. Let the exploration of C# programming and game development begin!

Source code example

```
using System;

namespace GuessTheNumber
{
    class Program
    {
        static void Main()
        {
            Console.WriteLine("iNetTutor.com");
            Console.WriteLine("Welcome to the Guess the Number game!");

            // Generate a random number between 1 and 100
            Random random = new Random();
            int targetNumber = random.Next(1, 101);

            // Initialize user guess and attempts
            int userGuess = 0;
            int attempts = 0;

            // Start the guessing loop
            while (userGuess != targetNumber)
            {
                // Prompt user for input
                Console.Write("Enter your guess (between 1 and 100): ");

                // Validate and parse user input
                if (int.TryParse(Console.ReadLine(), out userGuess))
                {
                    // Provide hints based on user's guess
                    if (userGuess < targetNumber)
                    {
                        Console.WriteLine("Too low. Try again!");
                    }
                    else if (userGuess > targetNumber)
                    {
                        Console.WriteLine("Too high. Try again!");
                    }

                    // Increment attempts
                    attempts++;
                }
                else
                {
                    Console.WriteLine("Invalid input. Please enter a valid number.");
                }
            }

            // Display success message
            Console.WriteLine($"Congratulations! You guessed the number {targetNumber} in {attempts} attempts.");
            Console.WriteLine("Please press any key in the keyboard to close the console.");
            Console.ReadKey();
        }
    }
}
```

Explanation

The provided source code is a C# program that implements a "Guess the Number" game. Let's go through the code and understand how it works.

Program Structure

The program is a console application with a single class named Program. It contains a Main method, which serves as the entry point of the program.

Game Setup

The program starts by displaying the messages "iNetTutor.com" and "Welcome to the Guess the Number game!" to the console.

Next, it generates a random number between 1 and 100 using the Random class. The generated number is stored in the targetNumber variable.

The program also initializes two variables: userGuess to store the user's guess and attempts to keep track of the number of attempts made by the user.

Guessing Loop

The program enters a while loop that continues until the user's guess matches the target number. Inside the loop, the program prompts the user to enter their guess by displaying the message "Enter your guess (between 1 and 100): ".

The user's input is then read using the Console.ReadLine method and stored in the userGuess variable. The program validates the user's input by using int.TryParse to check if the input can be successfully parsed as an integer. If the input is a valid number, the program proceeds to provide hints based on the user's guess.

If the user's guess is lower than the target number, the program displays the message "Too low. Try again!". If the guess is higher, the program displays "Too high. Try again!". In both cases, the attempts variable is incremented.

If the user's input is not a valid number, the program displays the message "Invalid input. Please enter a valid number."

Game Completion

Once the user's guess matches the target number, the program exits the while loop. It displays a success message using string interpolation to include the target number and the number of attempts made by the user.

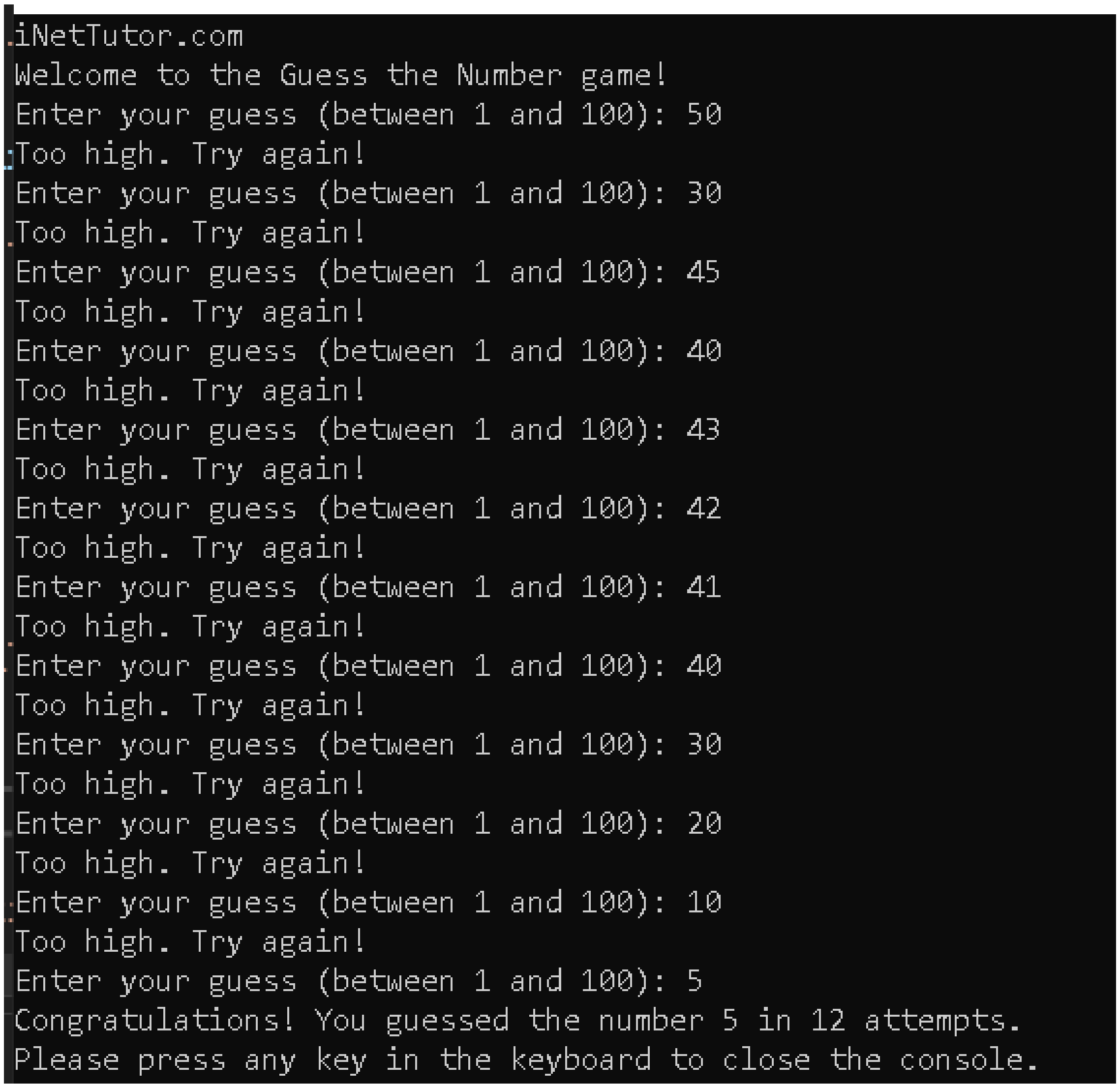
Finally, the program prompts the user to press any key to close the console using the Console.ReadKey method.

Example Execution

Here's an example execution of the program:

```
iNetTutor.com
Welcome to the Guess the Number game!
Enter your guess (between 1 and 100): 50
Too low. Try again!
Enter your guess (between 1 and 100): 75
Too high. Try again!
Enter your guess (between 1 and 100): 63
Too low. Try again!
Enter your guess (between 1 and 100): 70
Too high. Try again!
Enter your guess (between 1 and 100): 67
Congratulations! You guessed the number 67 in 5 attempts.
Please press any key on the keyboard to close the console.
```

Output



```
iNetTutor.com
Welcome to the Guess the Number game!
Enter your guess (between 1 and 100): 50
Too high. Try again!
Enter your guess (between 1 and 100): 30
Too high. Try again!
Enter your guess (between 1 and 100): 45
Too high. Try again!
Enter your guess (between 1 and 100): 40
Too high. Try again!
Enter your guess (between 1 and 100): 43
Too high. Try again!
Enter your guess (between 1 and 100): 42
Too high. Try again!
Enter your guess (between 1 and 100): 41
Too high. Try again!
Enter your guess (between 1 and 100): 40
Too high. Try again!
Enter your guess (between 1 and 100): 30
Too high. Try again!
Enter your guess (between 1 and 100): 20
Too high. Try again!
Enter your guess (between 1 and 100): 10
Too high. Try again!
Enter your guess (between 1 and 100): 5
Congratulations! You guessed the number 5 in 12 attempts.
Please press any key in the keyboard to close the console.
```

Summary

Congratulations on completing your journey through C# programming and creating a Guess the Number game! You've learned about random number generation, handling user input, and using loop structures. Let's summarize what you've learned and discuss what's next for you.

Key Takeaways

1. Random Number Generation: You learned how to create random numbers in C#. This is useful for making games more exciting and unpredictable.
2. User Input Handling: You learned how to handle the input from users. This means you can make sure they enter the right kind of information and give them helpful messages if they make a mistake.
3. Loop Structures: You used a loop called "while" to repeat actions in your game. This allows you to keep asking the user to guess the number until they get it right.

Next Steps

As you think about what you've accomplished, remember that this is just the beginning of your programming journey. Here are some ideas for what you can do next:

1. Keep Practicing: Practice regularly to get better at programming. Work on small coding exercises or projects to keep improving.
2. Try More Complex Projects: Challenge yourself by working on harder programming projects. This will help you understand C# better and improve your problem-solving skills.
3. Keep Learning: Stay up-to-date with new things in C# programming. Look for online resources, tutorials, and courses to learn more and expand your knowledge.
4. Join Programming Communities: Connect with other programmers online. Join forums or groups where you can share your projects and get feedback from others.

Meta Description

"Elevate your C# skills with Guess the Number exercises. Code optimization, user experience, and advanced techniques await! 🚀"