Rock, Paper, Scissors Game in C#

Introduction

In this lesson, we will be learning how to create a simple text-based Rock, Paper, Scissors game using the C# programming language. The game will involve asking the user to choose one of the options (rock, paper, or scissors), generating a random choice for the computer, and determining the winner based on the game rules.

This lesson is more than just building a game; it's about:

- Understanding fundamental programming concepts: Learn how to utilize random numbers, conditionals, and user input in a practical and engaging way.
- Developing your problem-solving skills: Break down the game logic, design your program's structure, and overcome any coding challenges you encounter.
- Having fun with C#: Discover the joy of creating something interactive and entertaining, igniting your passion for coding and its endless possibilities.

The Rock, Paper, Scissors game is a classic hand game played between two people, where each player simultaneously forms one of three shapes. The winner of the game is decided based on the following rules:

- Rock vs Paper -> Paper wins.
- Rock vs Scissors -> Rock wins.
- Paper vs Scissors -> Scissors wins.

To create this game in C#, we will need to take user input, generate a random choice for the computer, and compare the choices to determine the winner. We can use conditional statements and random number generation to implement the game logic.

Let's dive into the details of how to create this game in C#!

C# Implementation of the Rock, Paper, Scissors Game

To create the Rock, Paper, Scissors game in C#, we can follow these steps:

1. Take user input: Ask the user to choose one of the options (rock, paper, or scissors).
2. Generate a random choice for the computer: Use the Random class in C# to generate a random number representing the computer's choice.

3. Compare the choices: Use conditional statements to compare the user's choice and the computer's choice to determine the winner based on the game rules.

4. Display the result: Print the user's choice, the computer's choice, and the result of the game (e.g., "You win!", "Computer wins!", or "It's a tie!").

Objectives

We aim for participants to not only comprehend the fundamental dynamics of the game but also master the art of user input handling, sharpen their skills in implementing conditional statements, and apply the concept of randomization to create a dynamic and entertaining gameplay. Through understanding, learning, practicing, and applying these key elements, participants will emerge with a well-rounded proficiency in crafting interactive and enjoyable C# programs.

The main objectives of this lesson can be summarized as follows:

1. **Understand Game Dynamics:** Gain a comprehensive understanding of the rules and dynamics of the Rock, Paper, Scissors game. Explore how user input and randomization contribute to the unpredictability of the game.

2. **Learn User Input Handling:** Learn to efficiently handle and process user input in a C# console application. Understand the significance of validating and interpreting user choices for seamless gameplay.

3. **Practice Conditional Statements:** Practice implementing conditional statements in C# to determine the winner of each round based on the chosen options. Enhance your proficiency in decision-making within a program.

4. **Apply Randomization Techniques:** Apply randomization techniques to simulate the computer's choice in the game. Understand how randomness adds an element of unpredictability, making each game iteration unique.

Source code example

```
using System;

namespace RockPaperScissorsGame
{
    class Program
    {
        static void Main()
        {
            Console.WriteLine("iNetTutor.com");
            Console.WriteLine("Rock, Paper, Scissors Game");

            // Prompt user to choose rock, paper, or scissors
            Console.Write("Enter your choice (rock, paper, or scissors): ");
            string userChoice = Console.ReadLine().ToLower();

            // Validate user input
            if (userChoice == "rock" || userChoice == "paper" || userChoice == "scissors")
            {
                // Generate computer's random choice
                string[] options = { "rock", "paper", "scissors" };
                Random random = new Random();
                int randomIndex = random.Next(options.Length);
                string computerChoice = options[randomIndex];

                // Display choices
                Console.WriteLine($"Your choice: {userChoice}");
                Console.WriteLine($"Computer's choice: {computerChoice}");

                // Determine the winner
                if (userChoice == computerChoice)
                {
                    Console.WriteLine("It's a tie!");
                }
                else if ((userChoice == "rock" && computerChoice == "scissors") ||
                         (userChoice == "paper" && computerChoice == "rock") ||
                         (userChoice == "scissors" && computerChoice == "paper"))
                {
                    Console.WriteLine("You win!");
                }
                else
                {
                    Console.WriteLine("Computer wins!");
                }
            }
            else
            {
                Console.WriteLine("Invalid choice. Please enter rock, paper, or scissors.");
            }

            Console.ReadKey();
        }
    }
}
```

```csharp
using System;

namespace RockPaperScissorsGame
{
    class Program
    {
        static void Main()
        {
            Console.WriteLine("iNetTutor.com");
            Console.WriteLine("Rock, Paper, Scissors Game");

            // Prompt user to choose rock, paper, or scissors
            Console.Write("Enter your choice (rock, paper, or scissors): ");
            string userChoice = Console.ReadLine().ToLower();

            // Validate user input
            if (userChoice == "rock" || userChoice == "paper" || userChoice == "scissors")
            {
                // Generate computer's random choice
                string[] options = { "rock", "paper", "scissors" };
                Random random = new Random();
                int randomIndex = random.Next(options.Length);
                string computerChoice = options[randomIndex];

                // Display choices
                Console.WriteLine($"Your choice: {userChoice}");
                Console.WriteLine($"Computer's choice: {computerChoice}");

                // Determine the winner
                if (userChoice == computerChoice)
                {
                    Console.WriteLine("It's a tie!");
                }
                else if ((userChoice == "rock" && computerChoice == "scissors") ||
                         (userChoice == "paper" && computerChoice == "rock") ||
                         (userChoice == "scissors" && computerChoice == "paper"))
                {
                    Console.WriteLine("You win!");
                }
                else
                {
                    Console.WriteLine("Computer wins!");
                }
            }
            else
            {
                Console.WriteLine("Invalid choice. Please enter rock, paper, or scissors.");
            }

            Console.ReadKey();
        }
    }
}
```

Explanation

This implementation allows users to play the Rock, Paper, Scissors game in the console by interacting with the program.

We hope this explanation helps you understand the provided source code.

1. Namespaces and Classes:

- using System;: Imports the System namespace, providing essential tools for input/output, randomness, and string manipulation.
- namespace RockPaperScissorsGame: Creates a namespace to organize the code.
- class Program: Defines the main class containing the game's logic.

2. Entry Point:

- static void Main(): The starting point of the program's execution.

3. Introduction:

- Console.WriteLine("iNetTutor.com");: Prints a welcome message.
- Console.WriteLine("Rock, Paper, Scissors Game");: Displays the game's title.

4. User Input:

- Console.Write("Enter your choice (rock, paper, or scissors): ");: Prompts the user to enter their choice.
- string userChoice = Console.ReadLine().ToLower();: Reads the user's input, converts it to lowercase, and stores it in the userChoice variable.

5. Input Validation:

- if (userChoice == "rock" || userChoice == "paper" || userChoice == "scissors"): Checks if the user entered a valid choice.
    - If valid, proceeds to the game logic.
    - If invalid, displays an error message and ends the game.

6. Computer's Choice:

- string[] options = { "rock", "paper", "scissors" };: Creates an array of possible choices.

- Random random = new Random();: Creates a Random object to generate random numbers.
- int randomIndex = random.Next(options.Length);: Generates a random index within the options array.
- string computerChoice = options[randomIndex];: Retrieves the computer's choice from the array using the random index.

7. Display Choices:

- Console.WriteLine($"Your choice: {userChoice}");: Prints the user's choice.
- Console.WriteLine($"Computer's choice: {computerChoice}");: Prints the computer's choice.

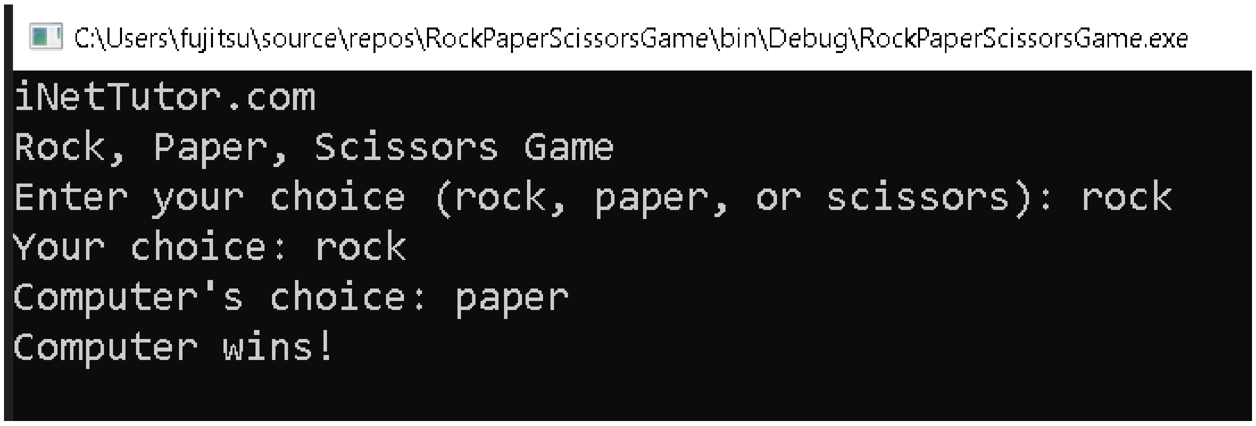
8. Determine Winner:

- Nested conditional statements: Determine the winner based on the game's rules:
  - Tie: If the choices are the same, it's a tie.
  - User wins: If the user's choice beats the computer's choice (based on the game's logic), the user wins.
  - Computer wins: Otherwise, the computer wins.

9. Pause:

- Console.ReadKey();: Waits for the user to press a key before closing the console window.

Output


C:\Users\fujitsu\source\repos\RockPaperScissorsGame\bin\Debug\RockPaperScissorsGame.exe

```
iNetTutor.com
Rock, Paper, Scissors Game
Enter your choice (rock, paper, or scissors): rock
Your choice: rock
Computer's choice: paper
Computer wins!
```

Summary

In this lesson, you embarked on a thrilling journey to create your very own Rock, Paper, Scissors game in C#. Armed with the power of code, you tackled four key skills:

- Understanding the game logic: Deciphering the winning and losing conditions that govern the game, ensuring accurate results for each round.
- Mastering C# concepts: Utilizing techniques like random number generation, conditional statements, and user input to bring the game to life.
- Crafting user interaction: Prompting players for their choices and displaying results in a clear and engaging way.
- Implementing core programming principles: Putting your new skills into practice by writing C# code step-by-step, refining and debugging your program along the way.

Remember, this is just the beginning! You can now apply these skills to build other interactive games or programs, pushing your boundaries and exploring the exciting world of C# coding. So, keep practicing, keep experimenting, and keep unleashing your creative potential!

Exercises and Assessment

To further improve the source code of the Rock, Paper, Scissors game, here are some suggested exercises and a lab exam:

1. Exercise: Error Handling - Modify the source code to include error handling for invalid user input. If the user enters an invalid choice, display an error message and prompt them to enter a valid choice (rock, paper, or scissors).
2. Exercise: Score Tracking - Enhance the game by adding a scoring system. Keep track of the number of wins, losses, and ties for the user and the computer. Display the scores after each round and declare the overall winner at the end of the game.
3. Enhance User Interaction - Modify the program to allow users to play multiple rounds without restarting the application. Keep track of the overall score and display it after each round.

These exercises and lab exam challenges will not only reinforce the core concepts covered in the lesson but also encourage learners to explore advanced programming techniques and enhance their problem-solving skills.

Meta Description

"Learn C# by creating a text-based Rock, Paper, Scissors game. Understand game rules, practice programming logic, and apply concepts in real-world scenarios."