Print Odd Numbers in CSharp

Introduction

In this lesson, we will learn how to write a C# program that prints odd numbers based on user input, with the input limit set to 100. The objective of our program is to provide a user-friendly interface that allows users to specify a number, and then display all the odd numbers up to that limit, each on a separate line.

To achieve this, we'll be utilizing two fundamental building blocks of C# programming:

- **Conditional Statements (if statement):** This acts like a decision-maker, checking if a certain condition is true. In our program, we'll use it to **validate your input** and ensure it falls within the acceptable range (between 1 and 100).
- **Loops (for loop):** This allows us to repeat a set of instructions a specific number of times. We'll employ a loop to efficiently **iterate through the desired range of odd numbers** based on your input.

Determining whether a number is odd can be accomplished by using the modulo operator (%). When a number is divided by 2, if the remainder is not zero, it indicates that the number is odd. We will leverage this property in our program to identify and print the odd numbers within the specified range.

By the end of this lesson, you will have a comprehensive understanding of how to utilize "if" statements, loops, and the modulo operator to print odd numbers based on user input. Learners will grasp the significance of conditionals and loops in problem-solving and gain proficiency in handling user input to produce desired outcomes.

Objectives

This lesson sets out to equip you with the essential skills to construct a C# program that tackles a specific task: printing odd numbers based on user input, with a maximum limit of 100.

Throughout this journey, you'll focus on four key objectives:

1. Understanding the Fundamentals:

- Grasp the core concepts of user input and its application in C# programs.
- Decipher the purpose and functionality of conditional statements (if statements) for making informed decisions based on provided data.
- Unravel the concept of loops (specifically, for loops) and their role in executing a set of instructions repeatedly.

2. Learning the Building Blocks:

- Gain practical knowledge of how to acquire user input in C#.
- Explore the syntax and structure of conditional statements (if statements) for implementing decision-making logic.
- Master the construction and utilization of for loops to control the program's flow and perform repetitive tasks efficiently.

3. Practicing the Techniques:

- Engage in hands-on exercises to solidify your understanding of user input, conditional statements, and loops.
- Apply the acquired knowledge to write and execute the C# program that prints odd numbers based on user input.
- Experiment with different scenarios and modify the program to explore its capabilities.

4. Applying the Knowledge:

- Recognize the broader applications of user input, conditional statements, and loops in C# programming.
- Leverage the gained skills to tackle more complex programming challenges that involve decision-making and repetitive tasks.
- Begin building your problem-solving repertoire by applying these fundamental concepts to real-world programming scenarios.

By actively pursuing these objectives, you'll not only construct the program to print odd numbers but also develop a strong foundation for your future endeavors in C# programming.

Source code example

```
1.  using System;
2.
3.  namespace PrintOddNumbers
4.  {
5.      class Program
6.      {
7.          static void Main(string[] args)
8.          {
9.              Console.WriteLine("iNetTutor.com - Print Odd Numbers");
10.             Console.WriteLine("Enter a number (up to 100):");
11.             int userInput = Convert.ToInt32(Console.ReadLine());
12.
```

```
            if (userInput <= 100 && userInput > 0)
            {
                Console.WriteLine($"Odd numbers up to {userInput}:");
                for (int i = 1; i <= userInput; i++)
                {
                    if (i % 2 != 0)
                    {
                        Console.WriteLine(i);
                    }
                }
            }
            else
            {
                Console.WriteLine("Invalid input. Please enter a number between 1 and 100.");
            }
            Console.ReadKey();
        }
    }
}
```

```
using System;

namespace PrintOddNumbers
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("iNetTutor.com - Print Odd Numbers");
            Console.WriteLine("Enter a number (up to 100):");
            int userInput = Convert.ToInt32(Console.ReadLine());

            if (userInput <= 100 && userInput > 0)
            {
                Console.WriteLine($"Odd numbers up to {userInput}:");
                for (int i = 1; i <= userInput; i++)
                {
                    if (i % 2 != 0)
                    {
                        Console.WriteLine(i);
                    }
                }
            }
            else
            {
                Console.WriteLine("Invalid input. Please enter a number between 1 and 100.");
            }
            Console.ReadKey();
        }
    }
}
```

Explanation

The provided source code is a beginner-friendly implementation of a C# program that prints odd numbers based on user input, with a limit of 100. Let's go through the code and understand its functionality:

1. Namespaces and Class:

- using System;: This line imports the System namespace, which contains essential functionalities like console input/output used in the program.
- namespace PrintOddNumbers: This line defines a namespace named PrintOddNumbers. Namespaces help organize code and prevent naming conflicts.
- class Program: This line declares a class named Program that contains the program's logic.

2. Main Method:

- static void Main(string[] args): This is the program's entry point where execution begins. The static keyword indicates the method belongs to the class itself, not a specific object. void signifies the method doesn't return any value. string[] args is an optional parameter array that can be used to pass arguments from the command line (not used in this case).

3. User Interaction:

- Console.WriteLine("iNetTutor.com - Print Odd Numbers");: Prints a message to the console displaying the program's purpose.
- Console.WriteLine("Enter a number (up to 100):");: Prompts the user to enter a number through the console.
- int userInput = Convert.ToInt32(Console.ReadLine());: Reads the user's input from the console, converts it to an integer using Convert.ToInt32(), and stores it in the userInput variable.

4. Input Validation (if statement):

- if (userInput <= 100 && userInput > 0): This if statement checks if the user's input is valid.
    - userInput <= 100: Ensures the input is less than or equal to 100 (upper limit).
    - userInput > 0: Ensures the input is positive (greater than 0).

5. Printing Odd Numbers (if statement block):

- This block of code executes only if the user enters a valid number (within the range of 1 to 100).

- Console.WriteLine($"Odd numbers up to {userInput}:");: Informs the user about the upcoming list of odd numbers up to the entered limit.
- for (int i = 1; i <= userInput; i++): This for loop iterates through the desired range of numbers.
  - int i = 1;: Initializes a loop counter i starting from 1.
  - i <= userInput: The loop continues as long as i is less than or equal to the user's input.
  - i++: The counter is incremented by 1 in each iteration.

6. Checking for Odd Numbers (inner if statement):

- if (i % 2 != 0): This if statement checks if the current number (i) is odd.
  - % is the modulo operator. It calculates the remainder when i is divided by 2.
  - != is the not equal operator. This condition checks if the remainder is not equal to 0. If true, the number is odd.

7. Printing Odd Numbers:

- Console.WriteLine(i);: Inside the inner if statement, if the number is odd (remainder is not 0), its value is printed to the console.

8. Handling Invalid Input (else block):

- This block executes if the user enters an invalid number (outside the range of 1 to 100).
- Console.WriteLine("Invalid input. Please enter a number between 1 and 100.");: Displays an error message informing the user about the invalid input.

9. Pausing the Console (Optional):

- Console.ReadKey();: This line pauses the console window until the user presses any key. This can be helpful for beginners to see the output before the console window closes.

Output

```
iNetTutor.com - Print Odd Numbers
Enter a number (up to 100):
30
Odd numbers up to 30:
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
```

Summary

In this lesson, we learned how to write a C# program that prints odd numbers based on user input, with a limit of 100. The key points covered in this lesson include:

1. Objective: The objective of the program is to print odd numbers based on user input, limiting the input to 100, with one number per line.
2. Input Validation: The program validates the user's input to ensure it is within the range of 1 to 100. If the input is invalid, an error message is displayed.
3. Looping: The program uses a for loop to iterate from 1 to the user's input. It checks each number to determine if it is odd using the modulo operator (%).

4. Conditional Check: The program uses an if statement with the condition i % 2 != 0 to check if the current number is odd. If it is, the number is printed on a new line.

5. Beginner-Friendly Code: The provided source code is beginner-friendly, with clear comments and explanations. It demonstrates the use of basic programming concepts such as input/output, loops, and conditional statements.

By understanding and applying these key points, beginners can successfully write a C# program to print odd numbers based on user input, limited to 100, with one number per line. This lesson serves as a foundation for further exploration and practice in C# programming.

Exercises and Assessment

To further improve the source code, you can focus on enhancing its code quality. Here's an exercise to help you practice code quality improvement:

1. Enhance User Experience:

- Modify the prompt message: Instead of just stating the upper limit, inform the user about the entire valid range (1 to 100).
- Implement input retry logic: If the user enters an invalid number, instead of just displaying an error message, provide them with a chance to re-enter a valid number within a set number of attempts.

2. Extend Functionality:

- Allow specifying both starting and ending numbers: Instead of limiting the user to entering just the upper limit, provide them with the flexibility to specify both the starting and ending numbers for the range of odd numbers they want to print.
- Calculate the sum of odd numbers: In addition to printing the odd numbers, calculate and display the sum of all the odd numbers within the specified range.

3. Advanced Error Handling:

- Implement exception handling: Instead of simply checking for the range using an if statement, explore using a try-catch block to handle potential exceptions that might arise during user input conversion (e.g., the user entering a non-numeric value).

By incorporating these suggestions, you can progressively improve the source code, making it more user-friendly, efficient, and capable of handling a wider range of scenarios. Remember to tailor the complexity of the exercises based on the target audience's skill level.

Quiz

1. Knowledge (Remembering):

What does the following line of code achieve in the program?

int userInput = Convert.ToInt32(Console.ReadLine());

a) Prints a message to the user.
b) Reads the user's input as a string.
c) Converts the user's input from a string to an integer.
d) Checks if the user's input is valid.

2. Comprehension (Understanding):

Why does the for loop in the program utilize the condition i <= userInput?

a) To ensure the loop iterates exactly 100 times.
b) To skip even numbers during iteration.
c) To repeat the loop as long as the counter i is less than or equal to the user's input.
d) To validate the user's input.

3. Application (Applying):

How can you modify the code to print only odd numbers within a specific range provided by the user (e.g., from 5 to 25)?

a) Change the loop condition to i < userInput.
b) Add an if statement inside the loop to check if the number is odd.
c) Modify the loop increment to i += 3.
d) Change the starting value of the loop counter i to the desired starting number (e.g., i = 5).

4. Analysis (Evaluating):

What potential issue could arise if the user enters a non-numeric value during input?

a) The program will automatically handle the error.
b) The loop might continue indefinitely.
c) The program will display an error message, but the code might crash.
d) The code might throw an exception due to the failed conversion from string to integer.

5. Synthesis (Creating):

Imagine you want to enhance the program to calculate the sum of all the odd numbers printed. How would you achieve this?

a) Modify the for loop to directly calculate the sum.
b) Introduce a new variable to store the sum and update it within the loop.
c) Add an if statement outside the loop to check for odd numbers and calculate their sum.
d) Utilize string manipulation techniques to process the printed odd numbers**.**

Meta Description

Learn to print odd numbers up to a user-defined limit in C#. Beginner-friendly code with explanation and key concepts.