Multiple Variable Value Checking

Check multiple values in a ForEach loop using a one line of code method by using logical operators.

This is a basic example in C# to iterate over a list and checking multiple conditions. In this example, the lambda expression n => { if (n % 2 == 0 && n > 5) { Console.WriteLine(n); } } checks if each number is even (n % 2 == 0) and greater than 5 (n > 5) using on one of code within the ForEach method.

Example Implementation

```
List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

numbers.ForEach(n =>
{
  if (n % 2 == 0 && n > 5) // Check if the number is even and greater than 5
  {
    Console.WriteLine(n);
  }
});
```

Additionally, you can check multiple conditions involving different types within a ForEach loop by using the appropriate type checks and casting or conversion as needed. This basic example shows how to handle different types in a ForEach loop.

Using object Type with Type Checking

Example Implementation

```
List<object> mixedList = new List<object> { 1, "hello", 3.14, 42, "world" };

mixedList.ForEach(item =>
{
  if (item is int intValue && intValue > 10) // check if item is an integer and greater than 10
  {
    Console.WriteLine($"Integer: {intValue}");
  }
  else if (item is string strValue && strValue.Length > 4) // check if item is a string and length is greater than 4
  {
    Console.WriteLine($"String: {strValue}");
  }
  else if (item is double doubleValue && doubleValue < 5.0) // check if item is a double and less than 5.0
  {
    Console.WriteLine($"Double: {doubleValue}");
  }
});
```

For this code, "item" is of type "object", so you need to use type checking (is keyword) and pattern matching to determine the actual type of "item" and then apply different conditions based on the "type".

Using a Custom Class with Multiple Properties

For a custom class with multiple properties check the properties within a ForEach loop. Each Person object is checked to see if their Age is greater than 25 and their Height is less than 6.0.

Example Implementation

```
public class Person
{
  public string Name { get; set; }
  public int Age { get; set; }
  public double Height { get; set; }
}

List<Person> people = new List<Person>
{
  new Person { Name = "Alice", Age = 30, Height = 5.5 },
  new Person { Name = "Bob", Age = 15, Height = 6.0 },
  new Person { Name = "Charlie", Age = 40, Height = 5.9 }
};

people.ForEach(person =>
{
  if (person.Age > 25 && person.Height < 6.0) // check if age is greater than 25 and height is less than 6.0
  {
    Console.WriteLine($"{person.Name} meets the criteria.");
  }
});
```

General Approach: The general approach when dealing with different types in a ForEach loop is to:

- Use Type Checking: Check the type of the object using is, as, or type conversion.
- Apply Type-Specific Logic: Once the type is confirmed, apply the conditions relevant to that type.
- Handle Casting and Conversions: Use casting or conversion to perform operations specific to the type.