C# Foreach Loop with Examples

In c#, the **Foreach** loop is useful to loop through each item in an array (/tutorial/csharp/csharp-arrays-with-examples) or collection (/tutorial/csharp/csharp-collections) object to execute the block of statements repeatedly.

Generally, in c# Foreach loop will work with the collection objects such as an array (/tutorial/csharp/csharp-arrays-with-examples), list (/tutorial/csharp/csharp-list), etc., to execute the block of statements for each element in the array (/tutorial/csharp/csharp-arrays-with-examples) or collection (/tutorial/csharp/csharp-collections).

After completing iterating through each element in the collection (/tutorial/csharp/csharp-collections), control will be transferred to the next statement following the **foreach** block.

In c#, we can use a break (/tutorial/csharp/csharp-break-statement-with-examples), continue (/tutorial/csharp/csharp-continue-statement-with-examples), goto (/tutorial/csharp/csharp-goto-statement-with-examples), and return (/tutorial/csharp/csharp-return-statement-with-examples) statements within the **foreach** loop to exit or continue to the next iteration of the loop based on our requirements.

Syntax of C# Foreach Loop

Following is the syntax of defining the **Foreach** loop in the c# programming language.

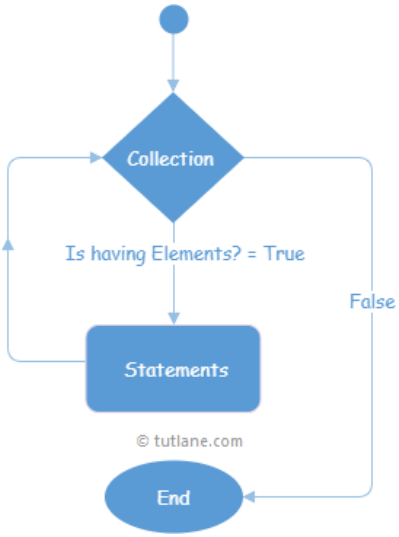
```
foreach (Type var_name in Collection_Object) {

// Statements to Execute

}
```

If you observe the above syntax, we defined a **foreach** loop with the collection object and required variable names to access the collection object's elements.

Here, **Type** is a built-in data-type (/tutorial/csharp/csharp-data-types-with-examples) or custom class type, and **var_name** is a variable (/tutorial/csharp/csharp-variables-with-examples) name to access elements from the collection object (**Collection_Object**) to use it in the foreach loop's body.

C# Foreach Loop Flow Chart

Following is the pictorial representation of the **foreach** loop process flow diagram in the c# programming language.



Now, we will see how to use the foreach loop in the c# programming language with examples.

C# Foreach Loop with Array Example

Following is the example of using a **foreach** loop in c# programming language to iterate or loop through array (/tutorial/csharp/csharp-arrays-with-examples) elements.

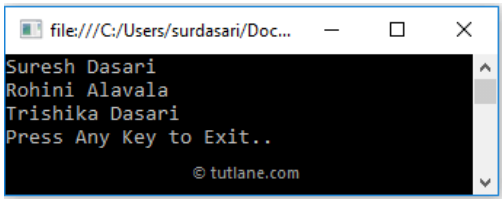
```
using System;

namespace Tutlane
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] names = new string[3] { "Suresh Dasari", "Rohini Alavala", "Trishika Dasari" };
```

```
            foreach (string name in names)
            {
                Console.WriteLine(name);
            }
            Console.WriteLine("Press Enter Key to Exit..");
            Console.ReadLine();
        }
    }
}
```

If you observe the above example, we created a string array object "**names**" and looped through each element of the array (/tutorial/csharp/csharp-arrays-with-examples) object using a **foreach** loop and assigning array elements to string variable "**name**".

To know more about arrays in the c# programming language, refer to c# arrays with examples (/tutorial/csharp/csharp-arrays-with-examples).

When we execute the above c# program, we will get the result below.



If you observe the above result, we loop through each element of the array (/tutorial/csharp/csharp-arrays-with-examples) and print those values on the console window based on our requirements.

C# Foreach Loop with List Example

Like c# **foreach** with arrays (/tutorial/csharp/csharp-arrays-with-examples), we can use the **foreach** loop with the list (/tutorial/csharp/csharp-list) object to process each element in the list (/tutorial/csharp/csharp-list) object, but inside the **foreach** loop, it won't allow us to modify (add or delete) the list (/tutorial/csharp/csharp-list) object items.

To learn more about lists in c# programming, refer to c# lists with examples (/tutorial/csharp/csharp-list).

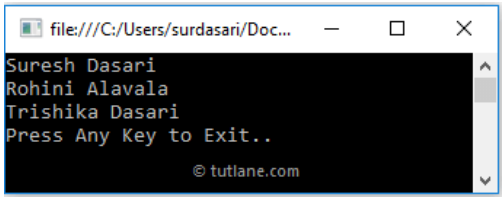
Following is the example of using a **foreach** loop in c# programming language to iterate or loop through list (/tutorial/csharp/csharp-list) elements.

```
using System;
using System.Collections.Generic;

namespace Tutlane
{
    class Program
    {
        static void Main(string[] args)
        {
            List names = new List() { "Suresh Dasari", "Rohini Alavala", "Trishika Dasari" };
            foreach (string name in names)
            {
                Console.WriteLine(name);
            }
            Console.WriteLine("Press Enter Key to Exit..");
            Console.ReadLine();
        }
    }
}
```

If you observe the above example, we used **System.Collections.Generic** namespace to access the **List** object and add string elements to the list (/tutorial/csharp/csharp-list). We used **foreach** to loop through items in the list to print them on the console window.

Output of C# Foreach Loop with List Example

When we execute the above c# program, we will get the result below.



This is how we can use the foreach loop in the c# programming language to loop through each element in the array (/tutorial/csharp/csharp-arrays-with-examples) or collection (/tutorial/csharp/csharp-collections) objects based on our requirements.

CONTACT US

Address: No.1-93, Pochamma Colony, Manikonda, Hyderabad, Telangana - 500089

Email: support@tutlane.com (mailto:support@tutlane.com)