TutorialKart

C# continue

C# continue

C# continue is used to skip further execution of statements for the current iteration in the loop and continue with the next iteration starting from condition evaluation.

The syntax of continue statement is:

```
continue;
```

Example : C# continue statement with for loop

In the following example, we use **continue** statement to skip the execution of statement(s) inside for loop after continue statement for this iteration. The loop continues normally with the updated value of `i` until the condition of for loop statement evaluates to false.

Program.cs

```
using System;

namespace CSharpExamples {
    class Program {
        static void Main(string[] args) {
            for(int i=0;i<10;i++){
                if(i==5){
                    continue;
                }
                Console.WriteLine(i);
            }
        }
    }
}
```

when `i` equals `5`, we continue the the for loop with updated value of `i`, by skipping the statements after continue statement for this iteration.

Output

```
PS D:\workspace\csharp\HelloWorld> dotnet run
0
1
```

```
2
3
4
6
7
8
9
```

Example : C# continue statement with while loop

In the following example, we use **continue** statement to the execution of the while loop during certain condition.

Program.cs

```
using System;

namespace CSharpExamples {
    class Program {
        static void Main(string[] args) {
            int i=0;
            while(i<10){
                if(i==5){
                    i++;
                    continue;
                }
                Console.WriteLine(i);
                i++;
            }
        }
    }
}
```

when `i` equals `5`, we increment `i` and continue with the the while loop with updated value of `i`, by skipping the statements after continue statement in the while loop.

If we do not increment `i` in the conditional if statement surrounding continue statement, the while loop becomes an infinite loop with `i` ever being `5`.

Output

```
PS D:\workspace\csharp\HelloWorld> dotnet run
0
1
2
3
4
6
7
8
9
```

Example : C# continue statement with foreach

In the following example, we use **continue** statement with foreach.

Program.cs

```
using System;

namespace CSharpExamples {
    class Program {
        static void Main(string[] args) {
            int[] nums = {2, 41, 15, 64};
            foreach(int num in nums){
                if(num==15){
                    continue;
                }
                Console.WriteLine(num);
            }
        }
    }
}
```

When `num` is equal to `15`, we are not executing the statement(s) after continue statement for this iteration. But the execution of foreach continues with the rest of values as usual.

Output

```
PS D:\workspace\csharp\HelloWorld> dotnet run
2
41
64
```

Note

- You can have multiple continue statements inside a loop for different conditions.
- When using continue statement, care has to be taken if the state of the loop is updated. Else, you may experience an infinite loop.

Conclusion

In this C# Tutorial, we learned to use C# continue statement to skip the execution of subsequent statements in the loop for a particular state of iteration.

C# Tutorial

✦ C# Tutorial

- C# Read Text File
- C# Write to File
- C# Delete File
- C# Copy File

C# Exception Handling

- C# try-catch
- C# finally
- C# throw
- C# Custom Exception
- C# SystemException
- C# DivideByZeroException
- C# NullReferenceException
- C# InvalidCastException
- C# IOException
- C# FieldAccessException

C# Collections

- C# List
- C# SortedList
- C# HashSet
- C# SortedSet
- C# Stack
- C# Queue
- C# LinkedList
- C# Dicionary
- C# SortedDictionary

C# Errors [Solved]

- C# Error: Class does not contain a constructor that takes arguments