Control Flow Loops in C#

Loops are fundamental constructs in programming that allow you to execute a block of code repeatedly based on a condition or a set of conditions. In C#, there are several types of loops: `while`, `do-while`, `for`, and `foreach`. Each type of loop serves a specific purpose and is suitable for different scenarios.

---

1. `while` Loop

Description

The `while` loop repeatedly executes a block of code as long as a specified condition is true. The condition is evaluated before each iteration of the loop.

Syntax

```
while (condition)
{
    // Code to execute while the condition is true
}
```

Example

```
using System;

class Program
{
    static void Main(string[] args)
    {
        int count = 0;

        while (count < 5)
        {
            Console.WriteLine($"Count is {count}");
            count++;
        }
    }
```

```
}
```

Explanation

- **Initialization**: The `count` variable is initialized to `0`.
- **Condition**: The loop continues as long as `count` is less than `5`.
- **Execution**: The `Console.WriteLine` method prints the current value of `count`.
- **Update**: The `count` variable is incremented by `1` in each iteration.

---

2. `do-while` Loop

Description

The `do-while` loop executes a block of code at least once before checking the condition. The condition is evaluated after each iteration.

Syntax

```
do
{
    // Code to execute at least once and while the condition is true
} while (condition);
```

Example

```
using System;
class Program
{
    static void Main(string[] args)
    {
        int count = 0;

        do
        {
            Console.WriteLine($"Count is {count}");
```

```
            count++;
        } while (count < 5);
    }
}
```

Explanation

- **Execution**: The `Console.WriteLine` method prints the current value of `count`.
- **Update**: The `count` variable is incremented by `1` in each iteration.
- **Condition**: The loop checks if `count` is less than `5` after executing the code block.

---

3. `for` Loop

Description

The `for` loop is used when the number of iterations is known beforehand. It provides a compact way to initialize a counter, specify a condition, and update the counter in a single line.

Syntax

```
for (initialization; condition; increment/decrement)
{
    // Code to execute while the condition is true
}
```

Example

```
using System;

class Program
{
    static void Main(string[] args)
    {
        for (int i = 0; i < 5; i++)
        {
            Console.WriteLine($"i is {i}");
```

```
    }
  }
}
```

Explanation

- **Initialization**: `int i = 0` initializes the loop counter.
- **Condition**: The loop continues as long as `i` is less than `5`.
- **Execution**: The `Console.WriteLine` method prints the current value of `i`.
- **Update**: `i++` increments the counter by `1` after each iteration.

---

4. `foreach` Loop

Description

The `foreach` loop is used to iterate over elements in a collection, such as arrays or lists. It simplifies the process of accessing each element without needing to manage an index variable.

Syntax

```
foreach (dataType item in collection)
{
  // Code to execute for each item in the collection
}
```

Example

```
using System;

class Program
{
  static void Main(string[] args)
  {
    string[] fruits = { "Apple", "Banana", "Cherry" };
```

```
        foreach (string fruit in fruits)
        {
            Console.WriteLine($"Fruit: {fruit}");
        }
    }
}
```

Explanation

- **Collection**: The `fruits` array contains a list of fruit names.
- **Iteration**: The `foreach` loop iterates over each `fruit` in the `fruits` array.
- **Execution**: The `Console.WriteLine` method prints the current fruit name.

Summary

- **`while` Loop**: Repeats execution as long as a condition is true. Condition is checked before each iteration.
- **`do-while` Loop**: Repeats execution at least once and continues as long as a condition is true. Condition is checked after each iteration.
- **`for` Loop**: Provides a compact way to initialize, conditionally execute, and update a counter. Suitable when the number of iterations is known.
- **`foreach` Loop**: Iterates over elements in a collection, simplifying the process of accessing each element without managing an index.

These loops are essential tools for controlling the flow of execution in a program and are used to automate repetitive tasks efficiently.