< Previous Next >

C# - Delegate

A function can have one or more parameters of different data types, but what if you want to pass a function itself as a parameter? How does C# handle the callback functions or event handler? The answer is - **delegate**.

A delegate is like a pointer to a function. It is a reference type data type and it holds the reference of a method. All the delegates are implicitly derived from `System.Delegate` class.

A delegate can be declared using **delegate** keyword followed by a function signature as shown below.

Delegate Syntax:

```
<access modifier> delegate <return type> <delegate_name>(<parameters>)
```

The following example declares a Print delegate.

Example: Declare delegate

```
public delegate void Print(int value);
```

The Print delegate shown above, can be used to point to any method that has same return type & parameters declared with Print. Consider the following example that declares and uses Print delegate.

Example: C# delegate

```
class Program
{
    // declare delegate
    public delegate void Print(int value);

    static void Main(string[] args)
    {
        // Print delegate points to PrintNumber
        Print printDel = PrintNumber;

        // or
        // Print printDel = new Print(PrintNumber);

        printDel(100000);
        printDel(200);

        // Print delegate points to PrintMoney
        printDel = PrintMoney;

        printDel(10000);
        printDel(200);
    }

    public static void PrintNumber(int num)
    {
        Console.WriteLine("Number: {0,-12:N0}",num);
    }

    public static void PrintMoney(int money)
    {
        Console.WriteLine("Money: {0:C}", money);
    }
}
```

Try it

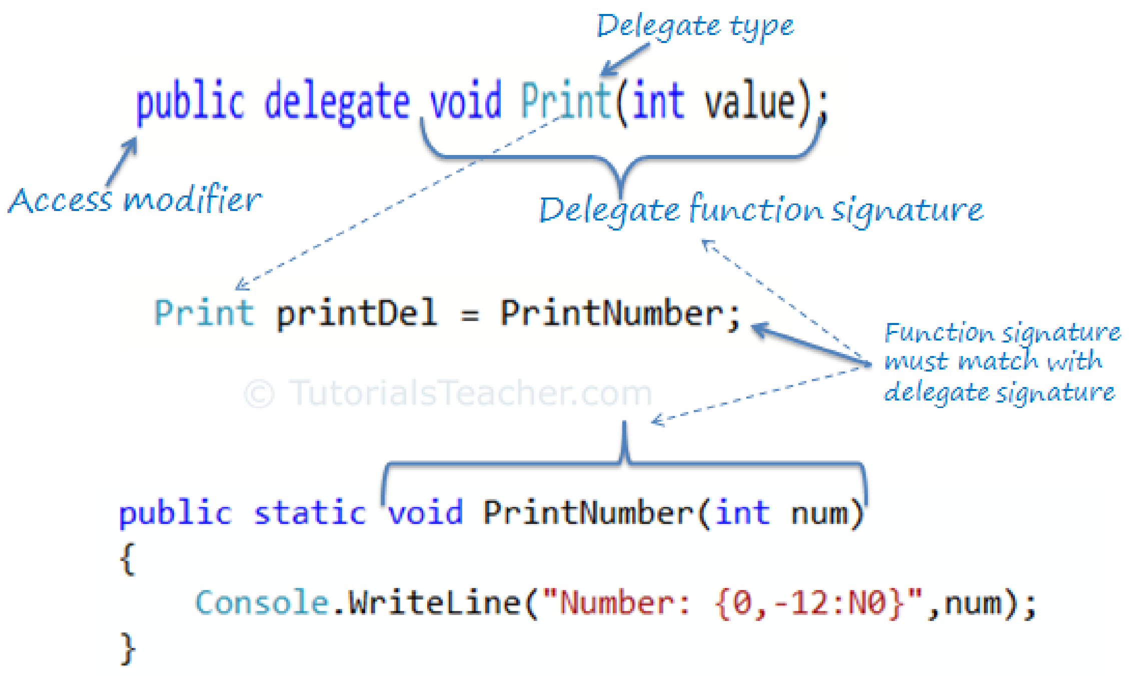
Output:

```
Number: 10,000
Number: 200
Money: $ 10,000.00
Money: $ 200.00
```

In the above example, we have declared Print delegate that accepts *int* type parameter and returns void. In the Main() method, a variable of Print type is declared and assigned a PrintNumber method name. Now, invoking Print delegate will in-turn invoke

PrintNumber method. In the same way, if the Print delegate variable is assigned to the PrintMoney method, then it will invoke the PrintMoney method.

The following image illustrates the delegate.


Delegate type
Access modifier
Delegate function signature
Function signature must match with delegate signature
© TutorialsTeacher.com

```
public delegate void Print(int value);

Print printDel = PrintNumber;

public static void PrintNumber(int num)
{
    Console.WriteLine("Number: {0,-12:N0}",num);
}
```

delegate in C#

Optionaly, a delegate object can be created using the `new` operator and specify a method name, as shown below:

```
Print printDel = new Print(PrintNumber);
```

Invoking Delegate

The delegate can be invoked like a method because it is a reference to a method. Invoking a delegate will in-turn invoke a method which id refered to. The delegate can be invoked by two ways: using () operator or using the Invoke() method of delegate as shown below.

Example: Invoking a delegate

```
Print printDel = PrintNumber;
printDel.Invoke(10000);

//or
printDel(10000);
```

```
Number: 10000
Number: 10000
```

Pass Delegate as a Parameter

A method can have a parameter of a delegate type and can invoke the delegate parameter.

Example: Delegate Parameter

```
public static void PrintHelper(Print delegateFunc, int numToPrint)
{
    delegateFunc(numToPrint);
}
```

In the above example, PrintHelper method has a delegate parameter of Print type and invokes it like a function: `delegateFunc(numToPrint)`.

The following example shows how to use PrintHelper method that includes delegate type parameter.

Example: Delegate parameter

```
class Program
{
    public delegate void Print(int value);


    static void Main(string[] args)
    {
        PrintHelper(PrintNumber, 10000);
        PrintHelper(PrintMoney, 10000);
    }

    public static void PrintHelper(Print delegateFunc, int numToPrint)
    {
        delegateFunc(numToPrint);
    }

    public static void PrintNumber(int num)
    {
        Console.WriteLine("Number: {0,-12:N0}",num);
    }

    public static void PrintMoney(int money)
    {
        Console.WriteLine("Money: {0:C}", money);
    }
}
```

Try it

Output:

```
Number: 10,000
Money: $ 10,000.00
```

Multicast Delegate

The delegate can points to multiple methods. A delegate that points multiple methods is called a multicast delegate. The "+" operator adds a function to the delegate object and the "-" operator removes an existing function from a delegate object.

Example: Multicast delegate

```
public delegate void Print(int value);

static void Main(string[] args)
{
    Print printDel = PrintNumber;
    printDel += PrintHexadecimal;
    printDel += PrintMoney;

    printDel(1000);

    printDel -=PrintHexadecimal;
    printDel(2000);
}

public static void PrintNumber(int num)
{
    Console.WriteLine("Number: {0,-12:N0}",num);
}

public static void PrintMoney(int money)
{
    Console.WriteLine("Money: {0:C}", money);
}

public static void PrintHexadecimal(int dec)
{
    Console.WriteLine("Hexadecimal: {0:X}", dec);
}
```

Try it

Output:

```
Number: 1,000
Hexadecimal: 3EB
Money: $ 1,000.00
Number: 2,000
Money: $2,000.00
```

As you can see in the above example, Print delegates becomes a multicast delegate because it points to three methods - PrintNumber, PrintMoney & PrintHexadecimal. So invoking printDel will invoke all the methods sequentially.

Delegate is also used with Event, Anonymous method, Func delegate, Action delegate.

Points to Remember :

1) Delegate is a function pointer. It is reference type data type.

2) Syntax: *public delegate void <function name>(<parameters>)*

3) A method that is going to assign to delegate must have same signature as delegate.

4) Delegates can be invoke like a normal function or Invoke() method.

5) Multiple methods can be assigned to the delegate using "+" operator. It is called multicast delegate.

Share

Tweet

Share

Whatsapp

< Previous

Next >

TUTORIALSTEACHER.COM

feedback@tutorialsteacher.com

TUTORIALS

- ASP.NET Core
- ASP.NET MVC
- IoC
- Web API
- C#
- LINQ
- Entity Framework
- AngularJS 1
- Node.js
- D3.js
- JavaScript

- jQuery
- Sass
- Https

E-MAIL LIST

Subscribe to TutorialsTeacher email list and get latest updates, tips & tricks on C#, .Net, JavaScript, jQuery, AngularJS, Node.js to your inbox.

Email address GO

We respect your privacy.

HOME PRIVACY POLICY TERMS OF USE ADVERTISE WITH US