C# Delegates

In C#, delegate is a *reference to the method*. It works like *function pointer* in C and C++. But it is objected-oriented, secured and type-safe than function pointer.

For static method, delegate encapsulates method only. But for instance method, it encapsulates method and instance both.

The best use of delegate is to use as event.

Internally a delegate declaration defines a class which is the derived class of **System.Delegate**.

C# Delegate Example

Let's see a simple example of delegate in C# which calls add() and mul() methods.

```
1. using System;
2. delegate int Calculator(int n);//declaring delegate
3.
4. public class DelegateExample
5. {
6.     static int number = 100;
7.     public static int add(int n)
8.     {
9.         number = number + n;
10.        return number;
11.    }
12.    public static int mul(int n)
13.    {
14.        number = number * n;
15.        return number;
16.    }
17.    public static int getNumber()
18.    {
19.        return number;
20.    }
21.    public static void Main(string[] args)
22.    {
23.        Calculator c1 = new Calculator(add);//instantiating delegate
24.        Calculator c2 = new Calculator(mul);
25.        c1(20);//calling method using delegate
26.        Console.WriteLine("After c1 delegate, Number is: " + getNumber());
27.        c2(3);
28.        Console.WriteLine("After c2 delegate, Number is: " + getNumber());
29.
30.    }
31. }
```

Output:

```
After c1 delegate, Number is: 120
After c2 delegate, Number is: 360
```