C# - ANONYMOUS METHODS

We discussed that delegates are used to reference any methods that has the same signature as that of the delegate. In other words, you can call a method that can be referenced by a delegate using that delegate object.

Anonymous methods provide a technique to pass a code block as a delegate parameter. Anonymous methods are the methods without a name, just the body.

You need not specify the return type in an anonymous method; it is inferred from the return statement inside the method body.

Writing an Anonymous Method

Anonymous methods are declared with the creation of the delegate instance, with a **delegate** keyword. For example,

```
delegate void NumberChanger(int n);
...
NumberChanger nc = delegate(int x)
{
   Console.WriteLine("Anonymous Method: {0}", x);
};
```

The code block *Console.WriteLine " AnonymousMethod: 0 " , x;* is the body of the anonymous method.

The delegate could be called both with anonymous methods as well as named methods in the same way, i.e., by passing the method parameters to the delegate object.

For example,

```
nc(10);
```

Example

The following example demonstrates the concept:

```
using System;

delegate void NumberChanger(int n);
namespace DelegateAppl
{
   class TestDelegate
   {
      static int num = 10;
      public static void AddNum(int p)
      {
         num += p;
         Console.WriteLine("Named Method: {0}", num);
      }

      public static void MultNum(int q)
      {
         num *= q;
         Console.WriteLine("Named Method: {0}", num);
      }

      public static int getNum()
      {
         return num;
      }
      static void Main(string[] args)
```

```
      {
         //create delegate instances using anonymous method
         NumberChanger nc = delegate(int x)
         {
            Console.WriteLine("Anonymous Method: {0}", x);
         };

         //calling the delegate using the anonymous method
         nc(10);

         //instantiating the delegate using the named methods
         nc =  new NumberChanger(AddNum);

         //calling the delegate using the named methods
         nc(5);

         //instantiating the delegate using another named methods
         nc =  new NumberChanger(MultNum);

         //calling the delegate using the named methods
         nc(2);
         Console.ReadKey();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result:

```
Anonymous Method: 10
Named Method: 15
Named Method: 30
```