



Feb 18th Assignment

By

Surya Teja Chandolu



1. Understand scope of variables in C#

The Scope of the variable determines the accessibility of the variable to a particular part of the application. Variables can be declared within the class, method, and code block of a loop, condition.

2. What are delegates in C#. Write the points discussed about delegates in the class. Write C# code to illustrate the usage of delegates.

A delegate is a type that represents references to methods with a particular parameter list and return type. When you instantiate a delegate, you can associate its instance with any method with a compatible signature and return type. In other words, a method must have the same return type as the delegate.

- A delegate is like a function pointer.
- Using delegates we can call to one or more methods.
- when declaring a delegate, return type and parameters must match with the methods you want to point using the delegate.
- Benefit of delegate is that, I using single call from delegate, all your methods pointing to delegate will be called.

Code:

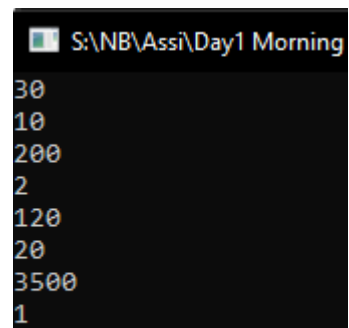
```
using System;

/*****
 * Author: Surya Teja
 * Purpose: Write C# code to illustrate the usage of delegates.
 * *****/

namespace DelegateE
{
    public delegate void Call(int a, int b);
    class Dele
    {
        public static void Add(int a, int b)
        {
            Console.WriteLine(a + b);
        }
        public static void Sub(int a, int b)
        {
            Console.WriteLine(a - b);
        }
        public static void Mul(int a, int b)
        {
            Console.WriteLine(a * b);
        }
        public static void Div(int a, int b)
        {
            Console.WriteLine(a / b);
        }
    }
    internal class Program
    {
        static void Main(string[] args)
```

```
    {  
        Call c = new Call(Dele.Add);  
        c += Dele.Sub;  
        c += Dele.Mul;  
        c += Dele.Div;  
  
        c(20, 10);  
        c(70, 50);  
  
        Console.ReadLine();  
    }  
}
```

Output:



S:\NB\Assi\Day1 Morning

30
10
200
2
120
20
3500
1

3. What are nullable types in C#. WACP to illustrate nullable types. Write some properties of nullable types (like HasValue)

The Nullable type allows you to assign a null value to a variable. Nullable types work with Value Type.

Properties:

- HasValue
- Value

Code:

```
using System;

/*****
 * Author: Surya Teja
 * Purpose: WACP to illustrate nullable types.
 * *****/

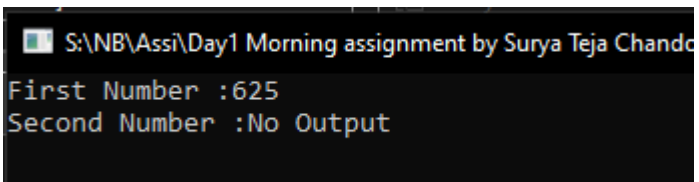
namespace NullableE
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int? FirstNumber = 25;
            int? SecondNumber = null;

            Console.Write("First Number :");
            if(FirstNumber.HasValue)
                Console.WriteLine(FirstNumber * FirstNumber);
            else
                Console.WriteLine("No Output");

            Console.Write("Second Number :");
            if (SecondNumber.HasValue)
                Console.WriteLine(SecondNumber * SecondNumber);
            else
                Console.WriteLine("No Output");

            Console.ReadLine();
        }
    }
}
```

Output:



```
S:\NB\Assi\Day1 Morning assignment by Surya Teja Chand
First Number :625
Second Number :No Output
```

4. Out, Ref - Parameters

Ref: The ref keyword passes arguments by reference. It means any changes made to this argument in the method will be reflected in that variable when control returns to the calling method.

Out: The out keyword passes arguments by reference. This is very similar to the ref keyword.

Code:

```
using System;

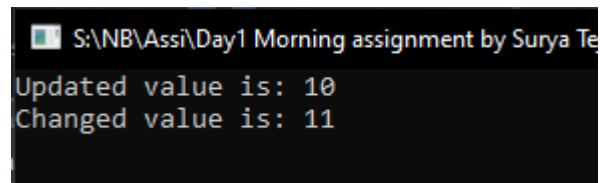
/*****
* Author: Surya Teja
* Purpose: Out and Ref.
* *****/

namespace RefAndOut
{
    class RefOut
    {
        public void update(out int a)
        {
            a = 10;
        }
        public void change(ref int d)
        {
            d = 11;
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            RefOut ro = new RefOut();
            int b;
            int c = 9;

            ro.update(out b);
            ro.change(ref c);
            Console.WriteLine($"Updated value is: {b}");
            Console.WriteLine($"Changed value is: {c}");

            Console.ReadLine();
        }
    }
}
```

Output:



```
S:\NB\Assi\Day1 Morning assignment by Surya Teja
Updated value is: 10
Changed value is: 11
```