ENUMS

In C#, an `enum` (short for enumeration) is a user-defined data type that has a fixed set of related values.
We use the `enum` keyword to create an enum. For example,

```
enum Months
{
   may,
   june,
   july,
}
```

Here,

- `Months` - enum name
- `may`, `june` and `july` - enum members (also known as string constants)

#define-an-enum Define an Enum

```
// define an enum named Weekdays
enum Weekdays
{
   sunday,
   monday,
   tuesday,
}
```

Here, we have defined an enum named `Weekdays`.

Access Enum Members

We use enum names along with `.` operator to access enum members.

Here from the above `Weekdays` enum,

- `Weekdays.sunday` - access the enum member, `sunday`
- `Weekdays.monday` - access the enum member, `monday`

Example: C# Enum

```
using System;

// define an enum
enum Weekdays
{
   sunday,
   monday,
   tuesday,
}

class Program
{
   static void Main()
   {
      // assign sunday to meetingDay
      Weekdays meetingDay = Weekdays.sunday;


      Console.WriteLine(meetingDay);
```

```
    }
}
```

Output

```
sunday
```

In the above example, we have defined an enum named `Weekdays`. Also, we have assigned the member value `sunday` to enum variable `meetingDay` as,

```
// assign sunday to meetingDay
Weekdays meetingDay = Weekdays.sunday;
```

C# Enum Values

In C#, we can assign numeric values to the enum members. For example,

```
using System;

// define an enum and assign numeric values
enum Season
{
    summer = 1,
    winter = 2,
    autumn = 3,
}
```

Here, we have assigned numeric values **1**, **2**, and **3** to the members summer, winter, and autumn respectively.

Enum Conversion

To print enum values, we need to convert enum members to its corresponding values using explicit type casting.
Let's see an example below,

```
using System;

//  an enum that contains shapes present in deck of card
enum Cards
{
   Diamond = 1,
   Spade = 2,
   Club = 3,
   Heart = 4,
}

class Program
{
   static void Main()
   {
      // type casting
      // convert string value "Spade" to integer value
      int myCard = (int)Cards.Spade;
```

```
        Console.WriteLine("Integer value of string constant is: " +
myCard);
    }
}
```

Output

```
Integer value of string constant is: 2
```

Here, we have converted the string value `"Spade"` to its corresponding integer value **2**.

Enum Default Values

If we have not assigned any value to the members of enum, by default **0** is assigned to the first member. Then the value of other members is increased by **1**. For example,

```
using System;

//  an enum that contains names of planet
enum Planet
{
    // value is 0
    mercury,
    // value is 1
    venus,
    // value is 2
    earth,

}
class Program
```

```
{
    static void Main()
    {
        // type casting enum to int
        int planet1 = (int)Planet.mercury;
        int planet2 = (int)Planet.venus;
        int planet3 = (int)Planet.earth;

        Console.WriteLine("Value of first member: " + planet1);
        Console.WriteLine("Value of second member: " + planet2);
        Console.WriteLine("Value of third member: " + planet3);
    }
}
```

Output

```
Value of first member: 0
Value of second member: 1
Value of third member: 2
```

In the above example, we have converted enum members to its corresponding numeric values by using typecast. Here,

- Mercury - **0 (first member)**
- Venus - **1 (second member)**
- Earth - **2 (third member)**

Note: We can assign different values to enum members. For example,

```
enum Planets
{
    mercury = 4,
    venus = 2,
    earth = 7,
```

```
}
```

Specifying Enum Type

In enum, the numeric value that we assign to the members can be of any of the integral numeric data types
like `byte`, `int`, `short`, `long`, `ushort`, or so on.
To specify the data type, we use `: typeName` after enum name. For example,

```
using System;
enum Holidays : long
{
   christmas = 123,
   thanksgiving = 124,
   halloween = 125,
}
```

In the above example, we have specified the data type of enum values as `enum Holidays : long`.