شمارش (Enumeration)

Enumeration یا شمارش راهی برای تعریف داده‌هایی است که می‌توانند مقادیر محدودی که شما از قبل تعریف کرده‌اید را بپذیرند. به عنوان مثال شما می‌خواهید یک متغیر تعریف کنید که فقط مقادیر جهت (جغرافیایی) مانند north، west، east و south را در خود ذخیره کند. ابتدا یک enumeration تعریف می‌کنید و برای آن یک اسم انتخاب کرده و بعد از آن تمام مقادیر ممکن که می‌توانند در داخل بدنه آن قرار بگیرند تعریف می‌کنید.

به نحوه تعریف یک enumeration توجه کنید:

```
enum enumName
{
   value1,
   value2,
   value3,
   .
   .
   .
   valueN
}
```

ابتدا کلمه کلیدی enum و سپس نام آن را به کار می‌بریم. در سی شارپ برای نامگذاری enumeration از روش پاسکال استفاده کنید. در بدنه enum مقادیری وجود دارند که برای هر کدام یک نام در نظر گرفته شده است. به یک مثال توجه کنید :

```
enum Direction
{
   North,
   East,
   South,
   West
}
```

در حالت پیشفرض مقادیری که یک enumeration می‌تواند ذخیره کند از نوع int هستند. به عنوان مثال مقدار پیشفرض north صفر و مقدار بقیه مقادیر یک واحد بیشتر از مقدار قبلی خودشان است. بنابراین مقدار east برابر ۱، مقدار south برابر 2 و مقدار west برابر 3 است. می‌توانید این مقادیر پیشفرض را به دلخواه تغییر دهید، مانند:

```
enum Direction
{
   North = 3,
   East = 5,
   South = 7,
   West = 9
}
```

اگر به عنوان مثال هیچ مقداری به یک عنصر اختصاص ندهید آن عنصر به صورت خودکار مقدار می کیرد.

```
enum Direction
{
   North = 3,
   East = 5,
   South,
   West
}
```

در مثال بالا مشاهده می‌کنید که ما هیچ مقداری به south در نظر نگرفته‌ایم بنابر این به صورت خودکار یک واحد بیشتر از east یعنی 6 و به west یک واحد بیشتر از south یعنی 7 اختصاص داده می‌شود. همچنین می‌توان مقادیر یکسانی برای عناصر enumeration در نظر گرفت. مثال :

```
enum Direction
{
   North = 3,
   East,
   South = North,
   West
}
```

می‌توانید مقادیر بالا را حدس بزنید؟ مقادیر north، east، south، west به ترتیب 3، 4، 3، 4 است. وقتی مقدار 3 را به north می‌دهیم مقدار east برابر 4 می‌شود. سپس وقتی مقدار southرا برابر 3 قرار دهیم به صورت اتوماتیک مقدار west برابر 4 می‌شود. اگر نمی‌خواهید که مقادیر آیتم‌های enumeration شما پیشفرض (از نوع int) باشد می‌توانید از نوع مثلاً byte به عنوان نوع داده‌ای آیتمهای آن اسفاده کنید.

```
enum Direction : byte
{
   North,
   East,
   South,
   West
}
```

نوع داده‌ای byte فقط شامل مقادیر بین 0 تا 255 می‌شود بنابر این تعداد مقادیر که شما می‌توانید به enumeration اضافه کنید محدود می‌باشد. به نحوه استفاده از enumeration در یک برنامه سی شارپ توجه کنید.

```
1  using System;
2
3  enum Direction
4  {
5      North = 1,
6      East,
7      South,
8      West
9  }
10
11 public class Program
12 {
13     public static void Main()
14     {
15         Direction myDirection;
16
17         myDirection = Direction.North;
18
19         Console.WriteLine("Direction: {0}", myDirection.ToString());
20     }
21 }
```

```
Direction: North
```

ابتدا enumeration را در خطوط 3-9 تعریف می‌کنیم. توجه کنید که enumeration را خارج از کلاس قرار داده‌ایم. این کار باعث می‌شود که enumeration در سراسر برنامه در دسترس باشد. می‌توان enumeration را در داخل کلاس هم تعریف کرد ولی در این صورت فقط در داخل کلاس قابل دسترس است.

```
class Program
{
    enum Direction
    {
        //Code omitted
    }

    static void Main(string[] args)
    {
        //Code omitted
    }
}
```

برنامه را ادامه می‌دهیم. در داخل بدنه enumeration نام چهار جهت جغرافیایی وجود دارد که هر یک از آنها با 1 تا 4 مقدار دهی شده‌اند. در خط 15 یک متغیر تعریف شده است که مقدار یک جهت را در خود ذخیره می‌کند. نحوه تعریف آن به صورت زیر است :

```
enumType variableName;
```

در اینجا enumType نوع داده شمارشی (مثلاً Direction یا مسیر) می‌باشد و variableName نیز نامی است که برای آن انتخاب کرده‌ایم که در مثال قبل myDirection است. سپس یک مقدار به متغیر myDirection اختصاص می‌دهیم (خط 17). برای اختصاص یک مقدار به صورت زیر عمل می‌کنیم :

```
variable = enumType.value;
```

ابتدا نوع Enumeration سپس علامت نقطه و بعد مقدار آن (مثلاً North) را می‌نویسیم. می‌توان یک متغیر را فوراً، به روش زیر مقدار دهی کرد :

```
Direction myDirection = Direction.North;
```

حال در خط ۱۹ با استفاده از Console.WriteLine() مقدار myDirection را چاپ می‌کنیم. توجه کنید که با استفاده از متد ToString() مقدار عددی myDirection را به رشته، جهت چاپ تبدیل می‌کنیم.

تصور کنید که اگر enumeration نبود شما مجبور بودید که به جای کلمات اعداد را حفظ کنید چون مقادیر enumeration در واقع اعدادی هستند که با نام مستعار توسط شما یا هر کس دیگر تعریف می‌شوند. متغیرهای شمارشی می‌توانند به انواع دیگری مانند int یا string تبدیل شوند. همچنین یک مقدار رشته‌ای می‌تواند به نوع شمارشی معادلش تبدیل شود.

تبدیل انواع شمارشی

می‌توان انواع شمارشی را به دیگر مقادیر تبدیل کرد و بالعکس. مقادیر شمارشی در واقع مقادیر عددی هستند که برای درک بهتر آنها، به هر عدد یک نام اختصاص داده شده است. به مثال زیر توجه کنید :

```
1  using System;
2
3  enum Direction
4  {
5      North,
6      East,
7      South,
8      West
9  }
10 public class Program
11 {
12     public static void Main()
13     {
14         Direction myDirection = Direction.East;
15         int myDirectionCode = (int)myDirection;
16
17         Console.WriteLine("Value of East is {0}", myDirectionCode);
18
19         myDirection = (Direction)3;
20         Console.WriteLine("nDirection: {0}", myDirection.ToString());
21     }
22 }
```

```
Value of East is 1

Direction: West
```

در خط 14 متغیر myDirection را به مقدار East نوع شمارشی Direction اختصاص داده‌ایم. در حالت پیشفرض مقدار East در داخل آیتمهای این داده شمارشی 1 می‌باشد. در خط 15 نشان نحوه تبدیل یک آیتم از نوع شمارشی به عدد صحیح معادل آن به روش تبدیل صریح نشان داده شده است. نحوه این تبدیل به صورت زیر است:

```
variable = (DestinationDataType)enumerationVariable;
```

از آنجاییکه متغیر myDirectionCode (خط 15) از نوع int است در نتیجه یک مقدار int باید در آن قرار بگیرد. می‌توان به سادگی نوع داده مقصد را در داخل یک جفت پرانتز قرار داد و آن را کنار نوع شمارشی بگذارید (خط 15). نتیجه یک مقدار تبدیل شده را برگشت می‌دهد. در خط 19 معکوس این کار را انجام می‌دهیم. در این خط یک مقدار صحیح را به یک مقدار شمارشی تبدیل می‌کنیم. مقدار 3 را برابر آیتم West قرار می‌دهیم.

برای تبدیل آن از روشی شبیه به تبدیل یک نوع شمارشی به صحیح استفاده می‌کنیم (تبدیل صریح). به این نکته توجه کنید که اگر عددی را که می‌خواهید تبدیل کنید در محدوده انواع شمارشی نباشد، تبدیل انجام می‌شود ولی آن آیتم شمارشی و عدد برابر هم نیستند. به عنوان مثال :

```
myDirection = (Direction)10;

Console.WriteLine("Direction: {0}", myDirection.ToString());
```

```
Direction: 10
```

از آنجاییکه عدد 10 مقدار هیچ کدام از آیتمهای نوع شمارشی مثال بالا نیست (مقدار آیتمهای نوع شمارشی مثال بالا به ترتیب 0 و 1 و 2 و 3 می‌باشد) خروجی Console خود عدد را نشان می‌دهد ولی اگر به جای عدد 10 هر کدام از مقادیر عددی ذکر شده را قرار دهید آیتم معادل با آن نمایش داده خواهد شد.

تبدیل یک نوع رشته‌ای به یک نوع شمارشی

می‌توان یک نوع رشته‌ای را به نوع شمارشی تبدیل کرد. مثلاً می‌خواهید رشته "West" را به نوع شمارشی Direction.West مثال بالا تبدیل کنید. برای این کار باید از کلاس Enum و فضای نام System به صورت زیر استفاده کنید :

```
Direction myDirection = (Direction)Enum.Parse(typeof(Direction), "West");

Console.WriteLine("Direction: {0}", myDirection.ToString());
```

```
Direction: West
```

متد ()Enum.Parse دارای دو پارامتر است. اولین پارامتر نوع شمارشی است. با استفاده از عملگر typeof نوع شمارشی را برگشت می‌دهیم. دومین پارامتر، رشته‌ای است که قرار است به نوع شمارشی تبدیل شود. چون مقدار برگشتی از نوعشی (object) است بنابراین یک تبدیل مناسب نوع شمارشی لازم است. با این جزییات الان می دانیم که چگونه یک رشته را به نوع شمارشی تبدیل کنیم.

```
enumType name = (enumType)Enum.Parse(typeof(enumType), string);
```

اگر رشته‌ای که به متد ارسال می‌کنید جز آیتمهای داده شمارشی نباشد با خطا مواجه می‌شوید.

ساختار (Struct)

ساختارها یا struct انواع داده‌ای هستند که توسط کاربر تعریف می‌شوند (user-define) و می‌توانند دارای فیلد و متد باشند. با ساختارها می‌توان نوع داده‌ای خیلی سفارشی ایجاد کرد. فرض کنید می‌خواهیم داده‌ای ایجاد کنیم که نه تنها نام شخص را ذخیره کند بلکه سن و حقوق ماهیانه او را نیز در خود جای دهد. برای تعریف یک ساختار به صورت زیر عمل می‌کنیم :

```
struct StructName
{
   member1;
   member2;
   member3;
   ...
   member4;
}
```

برای تعریف ساختار از کلمه کلیدی struct استفاده می‌شود. برای نامگذاری ساختارها از روش نامگذاری struct استفاده می‌شود. اعضا در مثال بالا (member1-5) می‌توانند متغیر باشند یا متد. در زیر مثالی از یک ساختار آمده است :

```
using System;

public struct Employee
{
    public string name;
    public int age;
    public decimal salary;
}

public class Program
{
    public static void Main()
    {
        Employee employee1;
        Employee employee2;

        employee1.name = "Jack";
        employee1.age = 21;
        employee1.salary = 1000;

        employee2.name = "Mark";
        employee2.age = 23;
        employee2.salary = 800;

        Console.WriteLine("Employee 1 Details");
        Console.WriteLine("Name  : {0}",   employee1.name);
        Console.WriteLine("Age   : {0}",   employee1.age);
        Console.WriteLine("Salary: {0:C}", employee1.salary);

        Console.WriteLine(); //Seperator

        Console.WriteLine("Employee 2 Details");
        Console.WriteLine("Name  : {0}",   employee2.name);
        Console.WriteLine("Age   : {0}",   employee2.age);
        Console.WriteLine("Salary: {0:C}", employee2.salary);
    }
}
```

```
Employee 1 Details
Name  : Jack
Age   : 21
Salary: $1000.00

Employee 2 Details
Name  : Mike
Age   : 23
Salary: $800.00
```

برای درک بهتر، کد بالا را شرح می‌دهیم. در خطوط 8-3 یک ساختار تعریف شده است. به کلمه Public در هنگام تعریف توجه کنید. این کلمه کلیدی نشان می‌دهد که Employee در هر جای برنامه قابل دسترسی و استفاده باشد و حتی خارج از برنامه. Public یکی از سطوح دسترسی است که توضیحات بیشتر در مورد آن در درسهای آینده آمده است. قبل از نام ساختار از کلمه کلیدی struct استفاده می‌کنیم. نام ساختار نیز از روش نامگذاری پاسکال پیروی می‌کند. در داخل بدنه ساختار سه فیلد تعریف کرده‌ایم. این سه فیلد مشخصات Employee (کارمند) مان را نشان می‌دهند. مثلاً یک کارمند دارای نام، سن و حقوق ماهانه می‌باشد. همچنین هر سه فیلد به صورت Public تعریف شده‌اند بنابراین در خارج از ساختار نیز می‌توان آنها را فراخوانی کرد. در خطوط 14 و 15 دو نمونه از ساختار Employee تعریف شده است. تعریف یک نمونه از ساختارها بسیار شبیه به تعریف یک متغیر معمولی است. ابتدا نوع ساختار و سپس نام آن را مشخص می‌کنید. در خطوط 17 تا 23 به فیلدهای مربوط به هر Employee مقادیری اختصاص می‌دهید. برای دسترسی به فیلدها در خارج از ساختار باید آنها را به صورت Public تعریف کنید. ابتدا نام متغیر را تایپ کرده و سپس علامت دات (.) و در آخر نام فیلد را می‌نویسیم.

وقتی که از عملگر دات استفاده می‌کنیم این عملگر اجازه دسترسی به اعضای مخصوص آن ساختار یا کلاس را به شما می‌دهد. در خطوط 25 تا 35 نشان داده شده که شما چطور می‌توانید به مقادیر ذخیره شده در هر فیلد ساختار دسترسی یابید. ساختارها انواع مقداری هستند. این بدین معنی است که اگر مثلاً در مثال بالا Employee2 را برابر Employee1 قرار دهید، Employee2 همه مقادیر صفات Employee1 را به جای اینکه به آنها مراجعه کند، کپی برداری می‌کند. کلاس یک ساختار ساده است ولی از انواع مرجع به حساب می‌آید. در مورد کلاس در درسهای آینده توضیح خواهیم داد. می‌توان به ساختار، متد هم اضافه کرد

```cs
using System;

public struct Employee
{
    public string name;
    public int age;
    public decimal salary;

    public void SayThanks()
    {
        Console.WriteLine("{0} thanked you!", name);
    }
}

public class Program
{
    public static void Main()
    {
        Employee employee1;
        Employee employee2;

        employee1.name = "Jack";
        employee1.age = 21;
        employee1.salary = 1000;

        employee2.name = "Mark";
        employee2.age = 23;
        employee2.salary = 800;

        Console.WriteLine("Employee 1 Details");
        Console.WriteLine("Name  : {0}",   employee1.name);
        Console.WriteLine("Age   : {0}",   employee1.age);
        Console.WriteLine("Salary: {0:C}", employee1.salary);

        employee1.SayThanks();

        Console.WriteLine(); //Seperator

        Console.WriteLine("Employee 2 Details");
        Console.WriteLine("Name  : {0}",   employee2.name);
        Console.WriteLine("Age   : {0}",   employee2.age);
        Console.WriteLine("Salary: {0:C}", employee2.salary);

        employee2.SayThanks();
    }
}
```

```
Employee 1 Details
Name  : Jack
Age   : 21
Salary: $1000.00
Jack thanked you!

Employee 2 Details
Name  : Mike
Age   : 23
Salary: $800.00
Mike thanked you!
```

مثال زیر اصلاح شده مثال قبل است.

در خطوط 9 تا 12 یک متد در داخل ساختار تعریف شده است. این متد یک پیام را در صفحه نمایش نشان می‌دهد و مقدار فیلد name را گرفته و یک پیام منحصر به فرد برای هر نمونه نشان می‌دهد. برای فراخوانی متد، به جای اینکه بعد از علامت دات نام فیلد را بنویسیم، نام متد را نوشته و بعد از آن همانطور که در مثال بالا مشاهده می‌کنید (خطوط 35 و 44) پرانتزها را قرار می‌دهیم و در صورتی که متد به آرگومان هم نیاز داشت در داخل پرانتز آنها را می‌نویسیم.