C# dilinde `struct` (yapı) değer tipi bir veri yapısıdır. `struct`, genellikle küçük ve hafif veri yapılarını temsil etmek için kullanılır ve nesnelerden (class) farklı olarak heap yerine stack üzerinde saklanır. `struct` tipleri, basit veri kapsüllerini tanımlamak için idealdir.

### Struct'ın Temel Özellikleri

1. \*\*Değer Tipi Olması\*\*: `struct` bir değer tipidir, yani stack üzerinde saklanır. Bu, kısa ömürlü ve küçük veri yapıları için idealdir.

2. \*\*Varsayılan Yapıcı\*\*: `struct`, varsayılan olarak parametresiz bir yapıcıya sahiptir ve tüm üyeler varsayılan değerlere atanır.

3. \*\*Inheritance Desteklememesi\*\*: `struct`, başka bir `struct` veya `class`'tan türetilemez ve `struct`'tan türetilemez. Ancak, arayüzleri (interface) uygulayabilir.

4. \*\*Boxing ve Unboxing\*\*: `struct`'lar `object` türüne dönüştürüldüğünde "boxing" işlemi gerçekleşir, bu da performans açısından maliyetli olabilir. Tersine "unboxing" de mümkündür.

### Struct Kullanımı

Aşağıda `struct`'ın nasıl tanımlandığını ve kullanıldığını gösteren bir örnek bulunmaktadır:

```csharp

using System;

public struct Point

{

public int X { get; set; }

public int Y { get; set; }

// Yapıcı (Constructor)

public Point(int x, int y)

{

X = x;

Y = y;

}

// Metot

public double DistanceTo(Point other)

{

int dx = X - other.X;

int dy = Y - other.Y;

return Math.Sqrt(dx \* dx + dy \* dy);

}

}

class Program

{

static void Main()

{

Point p1 = new Point(3, 4);

Point p2 = new Point(6, 8);

Console.WriteLine($"Point 1: ({p1.X}, {p1.Y})");

Console.WriteLine($"Point 2: ({p2.X}, {p2.Y})");

double distance = p1.DistanceTo(p2);

Console.WriteLine($"Distance between points: {distance}");

}

}

```

### Yapıcı ve Özellikler

- \*\*Yapıcı (Constructor)\*\*: `struct` yapısında yapıcılar tanımlayarak, veri üyelerini başlatabilirsiniz. Parametresiz varsayılan bir yapıcıyı içerir ve tüm üyeler varsayılan değerlere atanır.

- \*\*Özellikler (Properties)\*\*: `struct` içinde otomatik özellikler veya normal özellikler tanımlanabilir.

### Yapısal Farklılıklar ve Kullanım Alanları

- \*\*Bellek Alanı\*\*: `struct`'lar stack üzerinde saklanır, bu da daha hızlı erişim ve bellek yönetimi sağlar. Ancak, büyük veri yapıları veya sıkça kopyalanması gereken veriler için `class` kullanımı tercih edilir.

- \*\*İmmutability (Değişmezlik)\*\*: `struct`'lar genellikle küçük ve sabit veri yapıları için uygundur. `struct`'lar immutability (değişmezlik) sağlamak için kullanılabilir, bu da özellikle çoklu iş parçacığı (multi-threading) uygulamalarında faydalıdır.

### Ne Zaman `struct` Kullanılır?

- Küçük ve basit veri yapıları gerektiğinde

- Değişmez (immutable) veri yapıları gerektiğinde

- Performans açısından, stack üzerinde saklanması avantaj sağladığında

- Box ve unbox işlemlerinin performans maliyetine dikkat edilerek

Özetle, `struct`'lar hafif ve küçük veri yapılarını temsil etmek için kullanılır ve değer tipi olarak stack üzerinde saklanır. Bu özellikleri ile bazı senaryolarda `class`'lara göre daha uygun olabilirler. Ancak, yapısal farklılıkları ve performans etkileri göz önünde bulundurulmalıdır.