



YORUM CÜMLELERİ

Şimdi yaptığımız çalışmalar çok kısa ve basit olsa da, gerçek bir program çok uzun, karmaşık ve anlaşılması güç olabilir. Programcı bazen bir şey denemeye başlar, bitirir ve bitirdiği zaman da ne yaptığını unuttur. Bazen de bizim yazdığımız programı başkaları devam ettirmek durumunda olabilir. Diğer bir deyişle kodumuzun dokümantasyonunu yapmamız gerekebilir.

Değişkenleri isimlendirirken, anlamlı isimler seçmek gerektiğinden bahsetmişim. Ama sadece değişkenlerin anlamlı isimlere sahip olması yetmez. Bazen anlamlı isimler seçerken de çok açık, anlaşılır seçemediğimiz durumlarda işin içinden çıkılmaz olur.

Yalın sayılar kullanmaktan kaçınmak da kolalık sağlar. Mesela aşağıdaki örnek çok açık değildir:

```
1 | HaftalikUcret = 80 * GunSayisi;
```

Bunun yerine aşağıdaki gibi bir kullanım daha açıklayıcı olur:

```
1 | GunlukUcret = 80;  
2 | HaftalikUcret = GunlukUcret * GunSayisi;
```

Değişkenlerin anlamlı isimlere sahip olması ve tanımlanmamış sayıları kullanmaktan kaçınmak, programcıların kendi kendisinin dokümantasyonunu yapan programlar yazmak için kullandıkları yöntemlerdir.

Yorum cümleleri yazmak da kendi kendisinin dokümantasyonunu yapan program yazmanın güzel bir yoludur. Yorum cümleleri, derleyici tarafından dikkate alınmayan, ama programın üzerinde çalışacak olanlar için açıklayıcı bilgiler içeren, programcı tarafından programın yazıldığı esnada ya da daha sonra eklenen yazılardır.

Programcılar genellikle yorum satırları yazmak istemezler. Çünkü yorum satırları programın çalışması açısından hiçbir şey ifade etmez. Program bittikten sonra ve çalışmaya başladıktan sonra yorum satırları ekleyebilirsiniz. Fakat yazma aşamasındayken yorum satırı eklemek daha iyi bir yaklaşımdır. Zira, daha sonra unutacağınız bazı şeyler o anda aklınızdadır ve onları yazarsanız kaybolmamış olur.

C# iki çeşit yorum satırını destekler. Bunlardan birincisi tek satırlık yorum cümlesi denilen iki tane sağa eğik çizgi (//) kullanmaktır. İlk olarak C++ dilinde kullanılan ve daha C programcılarının da kullanmak istediği tek satırlık yorum cümlesi, daha sonra C diline de eklendi. Bu şekilde bir kullanımda satırın sonuna kadar olan her şey, derleyici tarafından yorum cümlesi olarak kabul edilir. C# da tek satırlık yorum cümlesi yapısını miras almıştır.

```
1 | int Ogrenciler = 32; //Bir sınıfın öğrenci mevcudu.
```

C#'ta genellikle satır sonları önemli değildir. Bir çok ifade tek satırda yazılabileceği gibi bir ifadeyi birden çok satıra bölmek de mümkündür. Bu yapı ifadeler arasında bırakılan fazladan boşluklar ya da diğer bir deyişle beyaz boşluklar gibidir. Tek satırlık yorum cümleleri ise bu kuralın dışındadır. Çünkü yorum cümlesi, satırın bittiği yerde sona erer. Bu durum C# dilinde, satır sonlarının diğer beyaz boşluklardan farklı değerlendirildiği çok nadir durumlardan biridir.

Diğer yorum cümlesi yazma şekli de sınırlanmış yorum cümlesidir. C'de uzun yıllar tek yorum cümlesi yazma şekli sınırlanmış yorum cümlesi oldu. /* karakterleri ile başlar, */ ile biter. Bir sınırlanmış yorum ifadesi birden çok satır tutabilir. /* karakterleri ile başlar, alt satırda da devam edebilir, */ karakterleri ile sona erer.

Yorum cümleleri derleyici tarafından hiçbir şekilde dikkate alınmaz.

- 1 | /* Bu satırda yorum cümlesi başladı,
- 2 | Alt satırda devam etti, hala devam ediyor, şimdi bitti */

Yorumun kendisi /* karakterlerini içerebilir, derleyici bunu kabul edebilir.

- 1 | /* Şimdi yorumun içinde /* karakteri kullandım, ama problem olmadı. */

Her iki yorum cümlesi türünü iç içe kullanabilirsiniz, ama bazen problem olur. Mesela aşağıdaki gibi bir kullanım yanlıştır:

- 1 | // Tek satırlık yorum cümlesi /* sınırlanmış yorum cümlesi,
- 2 | Sınırlanmış yorum cümlesinin devamı. */

Yukarıdaki yorum cümlelerinin kullanımının yanlış olmasının sebebi, her iki türün de anlamlarından kaynaklanıyor. // karakterleri, önünde bulunan satırın sonuna kadar tamamını yorum cümlesi kabul eder. Bu durumda alt satırda devam ettiğini düşündüğümüz yorum cümlesinin başlama karakterleri olan /*, hiç algılanmamış olur. Bu durumda alt satırdaki */ karakterlerinin daha önce başlamış bir yorumu bitirdiğini fark edemez. Hata olur. Oysa aşağıdaki kullanım doğrudur:

- 1 | /* sınırlanmış yorum cümlesi // Tek satırlık yorum cümlesi */

Derleyici // ifadelerini dikkate almaz, çünkü /* karakterleri ile başlan yorum cümlesi */ karakterlerine kadar devam eder. // karakterleri de yorumun bir parçası kabul edilir.

Bir çok programcı, yazdıkları kod sayfalarının en üstüne aşağıdaki gibi bir açıklayıcı ifade eklemeyi uygun bulur:

```
1 | /*****
2 | *
3 | * ExcelBenzeri.cs
4 | *
5 | * Yazan Yunus Özen, Mart 2006
6 | *
7 | * Bu Excel benzeri programı gördükten sonra Excel kullanmak istemezsiniz artık
8 | *
9 | *****/
```

Yukarıda görülen yorum cümleleri de standart /* karakterleri ile başlar, */ ile biter. Ama aradaki fazladan asteriksler estetik için koyulmuş, özel bir anlam ifade etmiyor.

Bunun gibi, yazdığımız her sınıfın veya yöntemin başına açıklama cümleleri ekleyebiliriz. Çok uzun bir program yazmışsak, programın başkaları tarafından anlaşılabilmesi böylece çok kolaylaşmış olur.

Yazdığımız ifadeleri de program içerisinde hemen bir üst satırlarına tek satırlık yorum cümlesi ekleyerek anlaşılır hale getirebiliriz.

- 1 | // Burada bir işçinin haftada kazanacağı parayı hesaplıyorum.

```
2 | HaftalikUcret = GunlukUcret * GunSayisi;
```

Değişkenlere isim verirken, anlamlı isimler seçmek ve düzgün kodlamak bazen yetmeyebilir. Mesela bir program bloğunu denedikten sonra, fikrimizi değiştirip başka bir blok denemek isteyebiliriz. Bu durumda ilk bloğu sınırlı yorum cümleleri haline getirip diğer bloğu yazabiliriz. Gerekirse de ikinci bloğu yorum cümleleri haline getirip ilk bloğu açarız.

```
1 | /*  
2 | GeldigiGunSayisi += 10;  
3 | KalanGun = ToplamGun - GeldigiGunSayisi;  
4 | */
```

Deneme yanılma yöntemi programlamada çoğu zaman işe yarar. Tecrübelerimiz de bize yol gösterir. Ama doğru programlamayı da öğrenmek gerekir.