</> C#



C#'a Giriş — Sözdizimi ve OOP

Programlamaya ilk adım: değişkenlerden nesne yönelimli programlamaya C# temelleri.

SowaCodeHK — Ders Notları
sowacodehk.com.tr · Hazar Kanat

Bu notta ne öğreneceksin? C#'ın ne olduğunu, değişken ve veri tiplerini, operatörleri, koşulları, döngüleri, metotları ve nesne yönelimli programlamanın (OOP) temel kavramlarını: sınıf, nesne, özellik, kapsülleme, kalıtım. Başlangıç seviyesidir; programlamaya ilk adımın olabilir.

C# Nedir?

C# (C-Sharp), Microsoft tarafından geliştirilen modern, güçlü ve okunması kolay bir programlama dilidir. Masaüstü uygulamaları, web siteleri (ASP.NET), oyunlar (Unity) ve daha fazlası C# ile yazılır. Güçlü tip sistemi ve nesne yönelimli yapısıyla büyük projeler için idealdir.

Değişkenler ve Veri Tipleri

Değişken, bir değeri saklayan kutudur. C#'ta her değişkenin bir **tipi** vardır (tam sayı mı, metin mi, ...):

```
int yas = 25;              // tam sayı
double boy = 1.78;         // ondalıklı sayı
string isim = "Hazar";     // metin
bool aktifMi = true;       // doğru/yanlış (true/false)
char harf = 'A';           // tek karakter
```

- `int` — Tam sayılar (1, 42, -7).
- `double` — Ondalıklı sayılar (3.14, 1.78).
- `string` — Metinler ("Merhaba").
- `bool` — Yalnızca true veya false.

> **İpucu:** Değişken adları anlamlı olsun: `int x` yerine `int yas` yaz. Kodun 6 ay sonra kendin bile daha kolay okursun.

Ekrana Yazdırma ve Operatörler

```
Console.WriteLine("Merhaba Dünya");

int a = 10;
int b = 3;
```

```
Console.WriteLine(a + b);   // 13  (toplama)
Console.WriteLine(a - b);   // 7
Console.WriteLine(a * b);   // 30
Console.WriteLine(a / b);   // 3   (tam bölme)
Console.WriteLine(a % b);   // 1   (kalan)
```

Koşullar (if / else)

Programın karar vermesini sağlar. "Eğer şu doğruysa bunu yap" mantığı:

```
int not = 75;

if (not >= 85) {
    Console.WriteLine("Pekiyi");
} else if (not >= 70) {
    Console.WriteLine("İyi");
} else {
    Console.WriteLine("Geçer");
}
```

Birden çok seçenek varsa `switch` daha temiz olur:

```
int gun = 3;
switch (gun) {
    case 1: Console.WriteLine("Pazartesi"); break;
    case 3: Console.WriteLine("Çarşamba"); break;
    default: Console.WriteLine("Diğer"); break;
}
```

Döngüler

Bir işi tekrar tekrar yapmak için döngü kullanılır. `for` belli sayıda tekrar için idealdir:

```
// 1'den 5'e kadar yaz
for (int i = 1; i <= 5; i++) {
    Console.WriteLine(i);
}
```

`while`, bir koşul doğru olduğu sürece çalışır:

```
int sayac = 0;
while (sayac < 3) {
    Console.WriteLine("Tekrar " + sayac);
```

```
    sayac++;
}
```

Metotlar (Fonksiyonlar)

Metot, bir işi yapan ve tekrar tekrar çağırabileceğin kod bloğudur. Kodun tekrarını önler:

```
// Metot tanımı
int Topla(int x, int y) {
    return x + y;
}

// Kullanımı
int sonuc = Topla(5, 3);   // sonuc = 8
Console.WriteLine(sonuc);
```

- `int` — Metodun geri döndüreceği değerin tipi (bir şey döndürmezse `void`).
- `Topla` — Metodun adı.
- `(int x, int y)` — Aldığı parametreler.
- `return` — Sonucu geri verir.

Nesne Yönelimli Programlama (OOP)

OOP, gerçek dünyadaki nesneleri koda taşıma yaklaşımıdır. Temel yapı taşı **sınıf (class)** ve ondan üretilen **nesnelerdir (object)**. Bir sınıf, bir şablondur; nesne ise o şablondan üretilmiş gerçek örnektir.

Sınıf ve Nesne

```
// Sınıf: bir şablon
class Araba {
    public string Marka;
    public int Hiz;

    public void Calistir() {
        Console.WriteLine(Marka + " çalıştı!");
    }
}

// Nesne: şablondan üretilmiş gerçek araba
Araba araba1 = new Araba();
araba1.Marka = "Toyota";
```

```
araba1.Hiz = 120;
araba1.Calistir();      // "Toyota çalıştı!"
```

- **Özellik (property/field)** — Nesnenin verileri: `Marka`, `Hiz`.
- **Metot** — Nesnenin yapabildikleri: `Calistir()`.
- `new` — Sınıftan yeni bir nesne üretir.

Kapsülleme (Encapsulation)

Verileri dış müdahaleden korumak için `private` yapıp, kontrollü erişim sağlanır. Böylece nesnenin iç durumu güvende kalır:

```
class Hesap {
    private double bakiye = 0;

    public void ParaYatir(double miktar) {
        if (miktar > 0) bakiye += miktar;
    }

    public double BakiyeGetir() {
        return bakiye;
    }
}
```

Kalıtım (Inheritance)

Bir sınıf, başka bir sınıfın özelliklerini devralabilir. Ortak kodu tekrar yazmazsın:

```
class Hayvan {
    public void Ye() { Console.WriteLine("Yemek yiyor"); }
}

// Kedi, Hayvan'dan miras alır
class Kedi : Hayvan {
    public void MiyavYap() { Console.WriteLine("Miyav!"); }
}

Kedi kedi = new Kedi();
kedi.Ye();          // miras aldı
kedi.MiyavYap();    // kendine ait
```

> **İpucu:** OOP'nin 3 temel prensibi: **Kapsülleme** (verini koru), **Kalıtım** (kodu miras al), **Çok biçimlilik** (aynı metot farklı davransın). İlk ikisi başlangıç için yeterli.

Özet

- C# güçlü tipli, modern bir dildir; web’den oyuna kadar her yerde kullanılır.
- Değişkenlerin tipi vardır: `int`, `double`, `string`, `bool`.
- Koşul (`if` / `switch`) ve döngü (`for` / `while`) ile program mantığı kurulur.
- Metotlar kod tekrarını önler.
- OOP: sınıf bir şablon, nesne ondan üretilen örnektir. Kapsülleme ve kalıtım temel taşlardır.
- Sıradaki adım: **ASP.NET Core** ile C#’ı web’e taşımak.