

Bu dokümanda anlatılan konularda bolca  
analoji (benzetme) yapılmıştır.  
Benzetmeler her zaman hatalıdır ve  
eksiktir. Ancak konuyu anlamamanızı sağlar.  
Buna dikkat etmeyi unutmayınız!

# Senkron Programlama

Programın giriş ve çıkış  
noktaları

Oğuzhan Karagüzel 20.10.2024

## İçindekiler

Şekiller Dizini .....	2
Tablolar Dizini .....	3
SENKRON – EŞZAMANLI ÇALIŞMA .....	4
PROGRAMIN SENKRON ÇALIŞMASI .....	4
PROGRAM.CS DOSYASI VE MAIN METODU .....	4
PROGRAMLAMA DİLLERİ VE SEVİYELERİ .....	5
SATIRLAR VE BLOKLAR .....	6
SENKRON PROGRAMLA ÖRNEĞİ .....	7
ÖDEV.....	8
TEST .....	9
Cevap Anahtarı .....	13

## Şekiller Dizini

Şekil 1 Program.cs Dosyası ve Main Metodu.....	5
Şekil 2. Satır ve blok bilgileri .....	6
Şekil 3 Tek satır program.....	7
Şekil 4 Sekronizasyon örneği 1.....	7
Şekil 5 Sekronizasyon örneği 2.....	8

Öğuzhan KARAGÜZEL

## Tablolar Dizini

Tablo 1. Promramlama dilleri ve Seviyeler .....	6
---	---

Öğuzhan KARAGÜZEL

## SENKRON – EŞZAMANLI ÇALIŞMA

“Eşzamanlı” tabirinin, “senkron” tabirine tam olarak karşılık olduğunu düşünmüyorum. Ancak şu an bundan daha iyi bir açıklama elimizde yok. Bu eğitim programında ilerledikçe yazdığınız kodların senkron çalışmasının ne demek olduğunu daha iyi anlayacaksınız. Belki “sıralı” tabiri daha uygun olacaktır. Ya da “Kodlar yukarıdan aşağıya doğru çalışır” olarak algılayacaksınız. Bende bu tabiri eğitimde sıklıkla kullanacağım.

Bu konuda anlaştığımızı var sayarak konumuza giriş yapalım. İşlemci ya da CPU (**Central Processing Unit** yani Türkçe olarak **Merkezi İşlem Birimi** anlamına gelir.) doğası gereği işlemleri ardışık olarak yapar. Yani sıradaki iş ne ise onu tamamlar ve ardından bir sonraki işe geçer. Bu davranış biçimi işlemciler için değişmez bir durumdur. Nasıl kod yazarsanız yazın, işlemci her zaman böyle davranacaktır.

Belki “Asenkron” ya da “Eşzamansız” programlama terimini duymuşsunuzdur. Biz buna sırasız programlama da diyebiliriz. Yazdığımız kodların asenkron çalıştığını söyler. Bu duyduğunuz doğrudur! Madem doğru, “İşlemciler her zaman senkron davranış gösterir” açıklamamız yanlış olacak! Hayır! Adı üstünde. **Asenkron programlama**. İşlemciler her zaman bir işi bitirip bir diğerine geçer. Ancak biz sıradaki işlerin sırasını değiştirebiliriz. Bu sayede işlemcinin asenkron çalıştığını gösteren bir illüzyon oluştururuz.

**Senkron:** İşlemin bitmesi beklenir. Bitince diğer işleme geçilir.

**Asenkron:** İşlemin bitmesi beklenmez. Bazı işlemlerin arka planda yürütülmesi sağlanır.

Eğitim programının ilerleyen aşamalarında asenkron programlamaya değineceğiz. Şimdilik asıl konumuzdan devam edelim.

## PROGRAMIN SENKRON ÇALIŞMASI

İşlemcinin “Senkron” çalışması ya da davranışından dolayı bizim kodlarımız her zaman senkron çalışmak zorundadır. Kodlarımızı her zaman bu davranışa göre yazmak zorundayız.

İşlemlerin sıralı olarak çalışması ya da yazdığımız kodların sıralı olarak çalışması konusunu anladığınızı düşünüyorum. Bu konuda anlaştıysak şimdi 2. Önemli konumuza geçelim. İşlemci, işlemleri sıralı olarak işleyecek. Bunun için bir yerden başlaması lazım. Peki nereden? İşlemlerimiz nereden başlayacak. Ayrıca yaşadığımız evrende sonsuz kavramının olmadığını da biliyoruz. Bu bizim sıralı işlemlerimiz ya da kodlarımız içinde geçerlidir. Yani sıraya alınan işlemlerin ya da kodlarımızın bir sonu olmak zorunda. Kısaca programımızın başlangıç ve bitiş noktaları olmak zorunda. Daha doğru bir terim ile programımızın bir giriş bir de çıkış noktası olmalı.

Programımızın giriş ve çıkış noktaları belirlenmiş, standardize edilmiş durumdadır. Hadi şimdi onlara bir bakış atalım.

## PROGRAM.CS DOSYASI VE MAIN METODU

Bir önceki derste belirtmiştim. Program.cs dosyası özel bir dosyadır. Main metodu ise özel bir metoddur.

```
internal class Program
{
    static void Main(string[] args)
    { //Programın giriş Noktası
        //Kodlarımız
    } //Programın çıkış noktası
}
```

Şekil 1 Program.cs Dosyası ve Main Metodu

Şekil 1’de programın giriş ve çıkış noktaları belirtilmiştir. Yazdığımız kodlar tam bu aralıkta, “Yukarıdan - aşağıya” satırlar (ya da bloklar) halinde çalışır.

Programımızın Yukarıdan aşağıya çalışacağını anladık. Ancak satır ya da blok ne demek! Şimdi okuyacaklarınız size biraz karmaşık ya da anlamsız gelecek. Oldukça basitleştirmeye çalışacağım. Bundan dolayı sadece ana noktaya odaklanmaya çalışınız.

Eğer yazılıma makina dilinden başlasaydınız şunu görecektiniz:

Önceki dersimizde gördüğünüz üzere “Console.WriteLine(“İlk projemi çalıştırdım”);” mesajını konsolumuza yazdırdık. Aslında yazdırmak ile kalmadık. Bu mesajı monitör sayesinde gözümüzle gördük. Sizin için çok basit ve tek satır olan bu işlem arka planda işlemci için belki de binlerce satır işleme denk geliyor. Henüz bu terimler için çok erken ama basitleştirirsek. Konsol görünümü için ekran kartına bilgi yolla. Konsolu ekranda göster. Mesajı oku, mesajı konsola yaz, görüntüyü işlemesi için ekran kartına gönder. Görüntüyü monitörde göster. Gibi gibi çok fazla işleme karşılık gelmektedir.

Eğer “Assembly” dilinden başlasaydınız bu işlem çok daha az kod tutacaktı. Çünkü bu dildeki kodların makine dilindeki karşılıkları bir bütün olarak yazılmış durumdadır. Bu sayede “Assembly” dilinde bir “satır” kod ile işlemciye, örneğin 15 tane işlemi yapmasını söylemiş oluyoruz.

Eğer “C” programlama dilinde yapsaydınız bu işlem belki de 2 satır tutacaktı.

Şimdi ise “C#” ile bu işlem sadece 1 satır tutmaktadır. Biraz genel kültür edinelim.

## PROGRAMLAMA DİLLERİ VE SEVİYELERİ

Seviye	Programlama Dilleri	Açıklama
Düşük Seviye	Makine Dili, Assembly	Donanımla doğrudan etkileşim sağlar, insan tarafından anlaşılması zordur. Performans açısından çok hızlıdır.
Orta Seviye	C, C++	Hem düşük hem de yüksek seviyeli dillerin özelliklerini taşır. Sistem programlaması ve donanım kontrolü için uygundur.
Yüksek Seviye	Python, Java, JavaScript, C#, PHP, Ruby	Kullanıcı dostudur ve insan tarafından daha kolay anlaşılır. Geliştirme süresi kısadır, ancak düşük seviyelere kıyasla

		performans açısından daha yavaş olabilir.
--	--	---

Tablo 1. Programlama dilleri ve Seviyeler

- 1. Düşük Seviye Programlama Dilleri:** Bu diller, donanım ile doğrudan iletişim kuran dillerdir. Makine dili (ikili kodlar) veya assembly dili bu kategoridedir. Bu diller, işlemci komutlarıyla doğrudan çalıştıkları için performans açısından çok hızlıdır ve donanım kontrolü gerektiren durumlarda kullanılır. Ancak, öğrenilmesi ve yazılması zordur, çünkü insan tarafından anlaşılması zordur.
- 2. Orta Seviye Programlama Dilleri:** Orta seviye diller, hem düşük hem de yüksek seviyeli dillerin özelliklerini bir araya getirir. Bu diller, sistem programlama ve donanım ile etkileşim kurmak için uygundur, fakat aynı zamanda soyutlama özellikleri de vardır. C ve C++ bu kategoriye girer. Donanımla daha yakından çalışmak isteyen ve aynı zamanda anlaşılabilir bir kod yazmak isteyenler için uygundur.
- 3. Yüksek Seviye Programlama Dilleri:** Bu diller, insan diline en yakın olan dillerdir. Kullanıcı dostu olup, gelişmiş soyutlamalar sağlar. Geliştiricilerin donanım detaylarını düşünmesine gerek kalmadan, uygulamalarını hızlı bir şekilde geliştirmelerine olanak tanır. Python, Java, JavaScript, C#, PHP ve Ruby gibi diller bu gruptadır. Bu diller, özellikle yazılım geliştirme süresinin hızlı ve verimli olmasını sağlar, ancak performans olarak düşük seviyelere göre genellikle daha yavaştır.

## SATIRLAR VE BLOKLAR

```
internal class Program
{
    static void Main(string[] args)
    { //blok başlangıcı ve sonraki satır başlangıcı
        Console.WriteLine("İlk projem"); //satır sonu ve sonraki satır başlangıcı
        Console.Read(); //satır sonu
    } //blok sonu
}
```

Şekil 2. Satır ve blok bilgileri

Şekil 2'ye dikkat edin. Programımızı bir blok içerisinde çalıştırıyoruz. Bloklar “{}” (Scope. Ben bunlara süslü parantez diyeceğim) ile belirtilir. Her süslü parantezin için bir blok koda denktir. Bir blok “{” ile açılır ya da başlar, “}” ile kapatılır ya da biter.

Blokları anladığınızı varsayarak şimdi satırlardan devam edelim.

Satırlar, Şekil 2'deki gibi, hali hazırda yazılmış bloklara işaret eden kodlardır. Satırlar ya “;” ile ya da “{” ile başlar. Ancak her zaman “;” ile biter. (C# ta bu her zaman böyledir.)

Peki neden böyledir? Çünkü bizim algımızda Programımız “Yukarıdan aşağıya doğru ya da satır satır” çalışır. Oysa bilgisayar için yukarı, aşağı ya da satır diye bir kavram bulunmamaktadır. Bizler “{}” ya da “;” işaretleri ile yukarıdan aşağıya doğru ya da satır satır kodlarız. Bilgisayara da bu işaretler ile kodlarımızı hangi sıra ile nasıl çalıştırılacağını söylemiş oluruz.

Bu mantıklıdır. Şekil 2'de ilk olarak konsola yaz dedik. Ardından konsoldan oku dedik. Aynı anda konsola yaz ve konsoldan oku diyemeyiz. Bu işlemcinin doğal davranış biçimine aykırı olurdu.

Şunu unutmayalım ki biz bir metin dosyasını düzenliyoruz. Bu metin dosyasındaki yazan yazılar bilgisayarın anlayacağı kodlara dönüştürülüyor. Bu dönüşümün sağlanması için kodlarımızın bir kurallar bütününe göre yazılması şarttır. C# programlama dilinde “{}” ya da “;” işaretlerini kullanırız. python gibi dillerde bu işaretlerin kullanımı zorunlu değildir. Ancak python programlama dilinde kodu alt satıra yazmak ya da belirli miktarda boşluk bırakmak zorunludur.

```
internal class Program
{
    static void Main(string[] args)
    { //blok başlangıcı ve sonraki satır başlangıcı
        Console.WriteLine("İlk projem"); Console.Read();
    } //blok sonu
}
```

Şekil 3 Tek satır program

Şekil 3’te gösterildiği gibi yazamazsınız. Python’da yazdığınız kodlar bu kurallara göre okunup bilgisayarın anlayacağı kodlara dönüştürülür.

C#’ta ise bu ayrımı “{}” ya da “;” işareti ile yaptığımızdan şekil 3’te ki gibi programımızı tek satır haline getirebiliriz.

Bu kurallar uyarak programımızı işlemcinin doğal davranış biçimi olan senkron çalışma biçimine uyumlu hale getirmiş oluruz. Metin dosyamız doğru bir biçimde işlenerek bilgisayarın anlayacağı kodlara dönüştürülür. Bu kodlar sorunsuz bir biçimde ardışık olarak işlemci tarafından çalıştırılır.

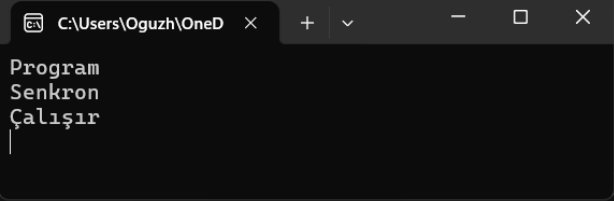
Günümüzde IDE’ler (entegre geliştirme ortamı - Integrated Development Environment) bu hataların önüne geçilecek biçimde kodlanmış programlardır. Sizde kodlarınızı bu programları kullanarak yazarsanız bu gibi hataların önüne geçmiş olursunuz. Bundan dolayı bu programları kullanmak oldukça önemlidir. Hatalarınızı daha en başından önler ve size inanılmaz bir hız kazandırır.

## SENKRON PROGRAMLA ÖRNEĞİ

Visual Studio’yu açalım ve programlamaya başlayalım. Burada kısa bir örnek gösterilip gerisi size ödev olarak verilecektir.

İlk derste verilen ödevlerden bir tanesinde sizlerden birden fazla mesajı ekrana yazdırmanız istenmiştir. Eğer o görevi başarı ile tamamladıysanız, şekil 4’te gördüğünüz kod size anlaşılır gelecektir.

```
internal class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Program");
        Console.WriteLine("Senkron");
        Console.WriteLine("Çalışır");
        Console.Read();
    }
}
```



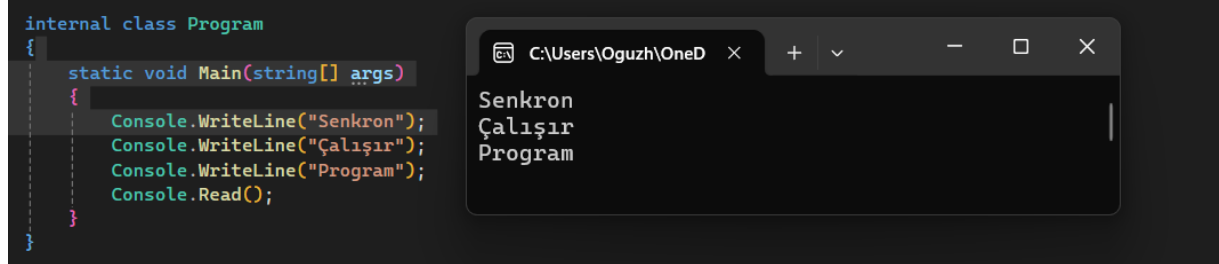
Şekil 4 Senkronizasyon örneği 1

Daha öncede belirttiğimiz üzere “Console.WriteLine();” metodu. Ona verilen mesajı konsola yazdıracaktır. Bu örnekte de bu metot kendinden beklendiği üzere görevini tamamlamıştır. Burada programımızın “Main” metodundan başlayacağını biliyoruz, “{” karakteri ise bizim başlangıç noktamız.



Bu durumda ilk yazdığım mesajın konsola ilk olarak yazılmasını beklerim. Şekil 4’te gördüğünüz üzere program tamda beklediğim gibi çalıştı.

Bu bilgi ışığında yeni bir test yapalım. Mesajlarımızın konumunu değiştirelim ve çıktıyı inceleyelim. Eğer kodlarımız, onları yazdığımız sıra ile çalışırsa doğru anladık demektir. Hadi deneyelim;



```
internal class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Senkron");
        Console.WriteLine("Çalışır");
        Console.WriteLine("Program");
        Console.Read();
    }
}
```

C:\Users\Oguzh\OneD x + - □ x

Senkron  
Çalışır  
Program

Şekil 5 Sekronizasyon örneği 2

Şekil 5’te de görüleceği üzere kodlarımız tam olarak;

- Yukarıdan aşağıya
- Blok – blok
- Satır – satır
- Adım adım

Hangi tabiri kullandığınızın bir önemi yok. Gördüğünüz gibi işlemcinin doğal davranış biçimine göre bizde kodlarımızı yazacağız. Olmasını istediğimiz işlemleri bu davranışa göre kodlayacağız. Eğer konsolda anlamlı bir mesaj görmek istiyorsanız mesajlarınızın anlamlı bir biçimde konsola yazılacağından emin olacak şekilde kod yazmalısınız.

Bu örneği temel alarak düşündüğünüzde bu sözler şu anlık size anlamsız geliyordur. Bu sözlerin önemini, eğitim programında ilerledikçe daha iyi anlayacaksınız. Bu kavramı şimdiden öğrenmek sizler için oldukça önemlidir.

## ÖDEV

İlk derste yeni bir projenin nasıl açılacağını öğrendiniz. Şimdi yine yeni bir proje açın. Aynı “console” uygulamasını .net framework 4.8 seçenekleri ile, adlandırma kurallarına uyarak oluşturun.

Ardından konsola 5 adet mesaj yazdırın. Çıktıyı görün. (Eğer konsolunuzu ekranda göremediyseniz, 1. Dersin dokümanını inceleyin! “Console.Read();”) Çıktıyı analiz edin ve programınızı kapatın. Ardından mesajlarınızın konumunu değiştirin. Programınızı tekrar çalıştırın ve çıktıyı görün. Çıktıyı analiz edin.

Aşağıda sizden istenileni yapmak zorunda değilsiniz. Ancak bunu sürekli olarak yapmak sizlere rapor ve doküman oluşturma yeteneği kazandıracaktır. Şuna emin olabilirsiniz ki bu yetenek sizi her alanda öne çıkaracaktır.

Elde ettiğiniz sonuçların kısa bir raporunu yazın.

Rapor yazımında dikkat edilmesi gerekenler:

- Başlık
- Ortam bilgileri. (sürüm ve araç hakkında bilgiler. Visual studio console app .net framework 4.8 gibi)

- Özet.
- Gövde (yapılan, beklenen, gözlemlenen)
- Sonuç.
- Yazar bilgisi

## TEST

Her Bir soru 5 puandır. Eğer test'ten 60 puanın altında aldıysanız. Dersi tekrar etmeniz faydanıza olacaktır.

### 1. Senkron programlamada işlemler nasıl gerçekleştirilir?

- a) Aynı anda yapılır
- b) Sıralı olarak yapılır
- c) Rastgele sıralanır
- d) Ters sırayla yapılır

Yanlış yaptıysanız, 'Senkron – Eşzamanlı programlama' başlığına bakın.

### 2. CPU'nun doğası gereği nasıl çalıştığı söylenir?

- a) Paralel çalışır
- b) Asenkron çalışır
- c) Ardışık olarak çalışır
- d) Rastgele işler yapar

Yanlış yaptıysanız, 'Senkron – Eşzamanlı programlama' başlığına bakın.

### 3. Senkron programlama için en uygun açıklama nedir?

- a) Kodlar yukarıdan aşağıya doğru sıralı olarak çalışır
- b) Kodlar aynı anda çalışır
- c) Kodlar işlemcinin hızına göre çalışır
- d) Kodlar sırasız çalışır

Yanlış yaptıysanız, 'Senkron – Eşzamanlı programlama' başlığına bakın.

### 4. Programın başlangıç ve bitiş noktası neresidir?

- a) Console.ReadLine() metodu
- b) Main metodu blokları
- c) Program.cs dosyası
- d) Bloklar arası

Yanlış yaptıysanız, 'Senkron – Eşzamanlı programlama' başlığına bakın.

**5. Asenkron programlama ile ilgili doğru ifade hangisidir?**

- a) İşlemcinin doğal çalışma biçimidir
- b) İşlemler sırayla yapılır
- c) İşlemlerin sırası değiştirilebilir
- d) Her zaman işlemciyi hızlandırır

Yanlış yaptıysanız, 'Senkron – Eşzamanlı programlama' başlığına bakın.

**6. Aşağıdakilerden hangisi düşük seviye programlama dillerine örnektir?**

- a) C#
- b) Python
- c) Assembly
- d) Java

Yanlış yaptıysanız, 'Programlama dilleri ve seviyeleri' başlığına bakın.

**7. Orta seviye programlama dillerinin bir avantajı nedir?**

- a) Donanımla doğrudan etkileşim sağlar
- b) Yüksek soyutlama seviyesine sahiptir
- c) Performans açısından yavaştır
- d) Sadece yüksek seviyede soyutlama sağlar

Yanlış yaptıysanız, 'Programlama dilleri ve seviyeleri' başlığına bakın.

**8. Yüksek seviye programlama dillerinde ne avantaj sağlar?**

- a) İnsan tarafından anlaşılması kolaydır
- b) Donanımla doğrudan etkileşim sağlar
- c) Makine diline çok yakındır

d) Performans açısından en hızlıdır

Yanlış yaptıysanız, 'Programlama dilleri ve seviyeleri' başlığına bakın.

**9. C# dilinde bloklar nasıl tanımlanır?**

a) Parantezler ile

b) Noktalı virgül ile

c) Süslü parantezler ile

d) İki nokta üst üste ile

Yanlış yaptıysanız, 'Satırlar ve Bloklar' başlığına bakın.

**10. Satır sonlandırıcı hangi sembole belirtilir?**

a) { }

b) ;

c) ( )

d) [ ]

Yanlış yaptıysanız, 'Satırlar ve Bloklar' başlığına bakın.

**11. Bir kod satırının sonunda noktalı virgül (;) olmadan yazılırsa ne olur?**

a) Program çalışmaya devam eder

b) Program hata verir

c) Kodlar doğru çalışır

d) Sadece uyarı verir

Yanlış yaptıysanız, 'Satırlar ve Bloklar' başlığına bakın.

**12. Senkron programlamanın temel özelliği nedir?**

a) İşlemler aynı anda yapılır

b) İşlemler sırayla yapılır

c) İşlemcinin hızına göre çalışır

d) İşlemler duruma göre yapılır

Yanlış yaptıysanız, 'Senkron Programlama' başlığına bakın.

**13. Console.WriteLine metodu ne yapar?**

- a) Mesajı bir dosyaya yazar
- b) Mesajı ekrana yazdırır
- c) Programı durdurur
- d) Kullanıcıdan girdi alır

Yanlış yaptıysanız, 'Senkron Programlama Örneği' başlığına bakın.

**14. Bir programdaki işlemlerin çalışma sırasını değiştirmek hangi programlama türü ile mümkündür?**

- a) Senkron
- b) Asenkron
- c) Statik
- d) Dinamik

Yanlış yaptıysanız, 'Senkron – Eşzamanlı Programlama' başlığına bakın.

**15. C# dilinde bloklar arasındaki işlemler nasıl yürütülür?**

- a) Aynı anda
- b) Sıralı olarak
- c) Rastgele
- d) Donanımın hızına bağlı olarak

Yanlış yaptıysanız, 'Satırlar ve Bloklar' başlığına bakın.

**16. IDE'lerin (Entegre Geliştirme Ortamı) temel avantajı nedir?**

- a) Kodu hızlı çalıştırması
- b) Yazım hatalarını önlemesi
- c) Kodun hızını artırması
- d) Donanım ile doğrudan etkileşim sağlaması

Yanlış yaptıysanız, 'Satırlar ve Bloklar' başlığına bakın.

**17. Senkron programlamanın işlemleri hangi sıraya göre gerçekleştirdiği söylenir?**

- a) Zaman sırasına göre
- b) Kodların yazıldığı sıraya göre

c) İşlemcinin kararına göre

d) Rastgele bir sıraya göre

Yanlış yaptıysanız, 'Senkron Programlama Örneği' başlığına bakın.

**18. Aşağıdakilerden hangisi yüksek seviye programlama dili değildir?**

a) JavaScript

b) Assembly

c) C#

d) Python

Yanlış yaptıysanız, 'Programlama dilleri ve seviyeleri' başlığına bakın.

**19. Aşağıdakilerden hangisi makine diline en yakın programlama dilidir?**

a) Python

b) Java

c) Assembly

d) Ruby

Yanlış yaptıysanız, 'Programlama dilleri ve seviyeleri' başlığına bakın.

**20. C#'ta bloklar hangi işaretlerle belirtilir?**

a) ()

b) {}

c) []

d) <>

Yanlış yaptıysanız, 'Satırlar ve Bloklar' başlığına bakın.

**Cevap Anahtarı**

1. b

2. c

3. a

4. b

5. c

6. c

7. a

8. a

9. c

10. b

11. b

12. b

13. b

14. b

15. b

16. b

17. b

18. b

19. c

20. b

Öğuzhan KARAGÜZEL