Bu dokümanda hemen her programlama dilinde bulunan if else yapılarına ve return ifadelerine değinilmiştir. Bu ifadeler programlamanın en temel kavramlarıdır ve neredeyse her dilde benzer şekilde yazılmaktadır. Hatta bu yapılar olmadan neredeyse düzgün çalışan bir program yapılamaz. Bundan dolayı bu yapılarda tecrübe kazanmak oldukça önemlidir.

KONTROL YAPILARI VE AKIŞ KONTROL

Oğuzhan Karagüzel 20.01.2025

İçindekiler

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

KONTROL YAPILARI

İF-ELSE YAPISINA GİRİŞ

Programlamada, karar verme yapıları bir kod parçasının belirli bir şarta bağlı olarak çalışmasını sağlar. C# dilinde bu karar yapıları if, else if ve else blokları ile uygulanır.

İf Yapısı

if anahtar kelimesi, bir şartı kontrol eder ve bu şart doğruysa (true) ilgili kodu çalıştırır.

```
if (kosul)
{
   // Koşul doğruysa bu kod çalışır.
   // Yanlışsa program olduğu gibi devam eder.
}
```

Else İf Ve Else Blokları

- else if: Birden fazla koşul kontrol etmeniz gerekiyorsa kullanılır.
- else: Hiçbir koşul doğru değilse çalışan bloktur.

```
if (kosul1)
{
   // kosul1 doğruysa bu kod çalışır.
   // Bu if bloğunun altında bulunan hiçbir else if ya da else bloğuna bakılmaz.
}
else if (kosul2)
{
   // Eğer kosul1 doğruysa bu kod bloğuna hiç bakılmaz. Eğer yanlışsa bu koşula bakılır.
   // kosul2 doğruysa bu kod çalışır.
   // Eğer kosul2 yanlışsa bu kod bloğu çalışmaz. Eğer varsa diğer else if bloklarına bakılır.
}
else
{
   // Yukarıdaki hiçbir koşul doğru değilse bu kod çalışır.
}
```

else Kullanmazsak Ne Olur?

else bloku kullanılmazsa, hiçbir şart doğru olmadığında program herhangi bir işlem yapmaz. else kullanımı, tüm durumların kapsandığını garanti etmek için faydalıdır.

BOOL VERİ TİPİNE GİRİŞ

Bool veri tipi, mantıksal değerleri (Şartları) saklamak için kullanılır. İki değer alabilir:

- true: Doğru
- false: Yanlış

bool Değerinin Kullanımı

bool tipi şaltları doğrudan kontrol etmek için kullanılabilir. Örneğin:

```
bool kosul = true;


if (kosul)
{
   Console.WriteLine("Koşul doğru!");
}
```

bool == true Neden Gereksizdir?

bool tipi zaten bir mantıksal ifade olduğundan, doğrudan kullanılabilir. Şöyle bir kod gereksizdir:

```
if (kosul == true)
{
   // Bu kod gereksiz yere uzatılmıştır.
}
```

Bunun yerine doğrudan:

```
if (kosul)
{
   // Daha kısa ve okunabilir.
}
```

Benzer şekilde, if (kosul == false) yerine if (!kosul) kullanılabilir.

Genellikle mantık konusunu iyi bilmeyen ya da acemi olan yazılımcılar “koşul == true” hatasına düşerler. Koşulun doğru kullanımı hem anlamayı kolaylaştırır hem de kod okunaklığını arttırır.

ÖRNEK PROGRAM: ÖĞRENCİNİN SINAVI GEÇİP GEÇMEDİĞİNİ KONTROL ETME

Bir öğrencinin sınavdan geçip geçmediğini kontrol eden bir program yazalım. Geçme notu 50 olsun.

```
static void Main()
{
    Console.Write("Sınav notunu giriniz: "); // kullanıcıdan notu istiyoruz.
    string not = Console.ReadLine(); // kullanıcının girdiği notu alıyoruz.metin olarak aldığımız için (rakam) int'e çevirmemiz gerekiyor.
    int parsedNot = int.Parse(not); // rakamı artık int'e çevirdik.

    if (parsedNot >= 50)
    {
        Console.WriteLine("Tebrikler, sınavı geçtiniz!");
    }
    else
    {
        Console.WriteLine("Maalesef, sınavı geçemediniz.");
    }
}
```

Şekil 1 Örnek Program. Öğrenci sınav sorgusu

Akış:

1. Kullanıcıdan bir not girmesi istenir.
2. Not >= 50 ise, program "Tebrikler, sınavı geçtiniz!" yazar.
3. Not 50'den küçükse, program "Maalesef, sınavı geçemediniz." yazar.

Bu örnek, if-else yapısının temel mantığını anlamanızı kolaylaştıracaktır.

Programın aynısını yazıp her iki durumu da test ediniz.

YENİ GELEN İSTEK!

Üniversitede profesörler bu programdan memnun olmadıklarını söylediler. Öğrencilerin harf notunu bilmesi gerektiğini ve bu programda girdiği notuna göre harf notunu öğrenmesini istiyorlar.

Profesörlerin ilettiğine göre harf notları şu şekilde veriliyormuş.

Tablo 1 Sınav sonuçlarına göre harf notları

SAYISAL NOT	HARF NOTU
100 = YA DA > NOT = YA DA > 90	A
89 = YA DA > NOT = YA DA > 80	B
79 = YA DA > NOT = YA DA > 70	C
69 = YA DA > NOT = YA DA > 60	D
59 = YA DA > NOT = YA DA > 50	E
50 > NOT	F (KALDI)

Bu yeni gelen istek doğrultusunda programımızı geliştirmemiz gerekecektir. Tablodan anlaşılacağı üzere else if yapısını kullanmak zorundayız.

```
static void Main()
{
    Console.Write("Sınav notunu giriniz: ");
    int not = int.Parse(Console.ReadLine());

    if (not >= 90 && not <= 100)
    {
        Console.WriteLine("Harf Notu: A");
    }
    else if (not >= 80 && not <= 89)
    {
        Console.WriteLine("Harf Notu: B");
    }
    else if (not >= 70 && not <= 79)
    {
        Console.WriteLine("Harf Notu: C");
    }
    else if (not >= 60 && not <= 69)
    {
        Console.WriteLine("Harf Notu: D");
    }
    else if (not >= 50 && not <= 59)
    {
        Console.WriteLine("Harf Notu: E");
    }
    else if (not < 50 && not >= 0)
    {
        Console.WriteLine("Harf Notu: F (Kaldı)");
    }
    else
    {
        Console.WriteLine("Geçersiz bir not girdiniz. Lütfen 0 ile 100 arasında bir değer giriniz.");
    }
}
```

Şekil 2 Yeni gelen istek ile geliştirilen program

Bu resmi yapay zekaya verip “bu programı yazabilir misin?” gibi işler yapmayın. Teker teker yazın. Bu hem kas hafızanızı hem bool kullanımını hem de şart kullanım yeteneğinizi arttıracaktır.

DİKKAT

Programımızı sadece gelen istek doğrultusunda değil. Olası hatalara göre de güçlendirmiş olduk.

“else” kullanımına dikkat ediniz. Bir önceki programda eğer kullanıcı 101 ya da 500 gibi notlar girseydi yine geçer not alacaktır. Ya da 49 yerine yanlışlıkla 498 yazmış olsaydı yine geçer not alacaktı. Burada ise girilen notun 100 ‘den küçük olacağını da garanti etmiş olduk. Ancak negatif sayılar için hala daha bir kontrol yapmamış olduk.

Artık orası size ödev.

TEK SATIR KULLANIM.

Daha sonra bu durumu döngülerde de göstereceğim. Ancak şimdi kontrol bloklarında bakalım.

Bildiğiniz üzere bir if bloğu şu şekilde yazılmaktadır.

İf(kosul)

{

Control.WriteLine(“Koşul sağlandı”);

}

Ancak şöyle bir kısaltma yapabiliriz.

İf (kosul)

Control.WriteLine(“Koşul sağlandı”);

Ya da

İf (kosul) Control.WriteLine(“Koşul sağlandı”);

Eğer bloğumuz tek satır ise süslü parantez “{}” kullanmanız gerekmemektedir.

Bu bilgi ile programımızı tekrar güncelleştirelim

```
static void Main()
{
    Console.Write("Sınav notunu giriniz: ");
    int not = int.Parse(Console.ReadLine());

    if (not >= 90 && not <= 100) Console.WriteLine("Harf Notu: A");
    else if (not >= 80 && not <= 89) Console.WriteLine("Harf Notu: B");
    else if (not >= 70 && not <= 79) Console.WriteLine("Harf Notu: C");
    else if (not >= 60 && not <= 69) Console.WriteLine("Harf Notu: D");
    else if (not >= 50 && not <= 59) Console.WriteLine("Harf Notu: E");
    else if (not < 50 && not >= 0) Console.WriteLine("Harf Notu: F (Kaldı)");
    else Console.WriteLine("Geçersiz bir not girdiniz. Lütfen 0 ile 100 arasında bir değer giriniz.");
}
```

Şekil 3 Tek satır kontrol bloğu kullanımı.

TERNARY IF YAPISINA GİRİŞ

Ternary if, kısa bir şekilde karar verme yapısını ifade etmeye yarar. if-else yapısının tek satırda yazılabilmesini sağlar. Genel yapı şu şekildedir:

koşul ? doğruysa_bu : yanlışsa_bu;

- koşul: Kontrol edilen mantıksal ifade.
- doğruysa_bu: Şart doğruysa çalışacak ifade.
- yanlışsa_bu: Şart yanlışsa çalışacak ifade.
- ? : koşulu kontrol edeceğimizi belli ederiz.
- “:” bu ifadenin sol tarafı eğer koşu doğru ise yapılacak olan. Sağ tarafı eğer koşul yanlış ise yapılacak olandır.

Örnek:

```
static void Main()
{
    int not = 85;
    string sonuc = not >= 50 ? "Geçti" : "Kaldı";
    Console.WriteLine(sonuc);
}
```

Şekil 4 Ternary if yapısı kullanımı

Bu kodda, not 50'den büyük veya eşit ise "Geçti" yazdırılır; aksi halde "Kaldı" yazdırılır.

Kullanıcı Notu Kontrolü (Ternary If İle)

Aşağıdaki örnekte, kullanıcıdan alınan notun geçerli bir aralıkta olup olmadığını kontrol eden ve harf notunu yazdıran bir program yazacağız. Eğer not 0-100 arasında değilse, hata mesajı gösterilecektir.

Burada sıkıntı şudur. Eğer doğru ise doğru sonuç döndürülür. Eğer yanlış ise tekrardan kontrol yapılır.

kosul ? doğru : (kosul2 ? doğru : (kosul3 ? doğru : (……)));

şeklinde dilediğince uzatılabilir. Ancak bu bazı durumlarda okumayı oldukça zorlaştırmaktadır. Dikkatlice kullanılmalıdır. Ayrıca henüz öğrenim aşamasında olduğunuz için bu tarz iç içe kullanımdan kaçının. Bu sizi oldukça zorlayacaktır.

Ben yine de incelemeniz için aşağıya kod bloğunu bırakıyorum.

```
static void Main()
{
    Console.Write("Sınav notunu giriniz: ");
    int not = int.Parse(Console.ReadLine());

    // Notun geçerli aralıkta olup olmadığını kontrol et
    string mesaj = (not >= 0 && not <= 100)
        ? (not >= 90 ? "Harf Notu: A"
            : not >= 80 ? "Harf Notu: B"
            : not >= 70 ? "Harf Notu: C"
            : not >= 60 ? "Harf Notu: D"
            : not >= 50 ? "Harf Notu: E"
            : "Harf Notu: F (Kaldı)")
        : "Geçersiz bir not girdiniz. Lütfen 0 ile 100 arasında bir değer giriniz.";

    Console.WriteLine(mesaj);
}
```

Şekil 5 Ternary if ile not kontrolü programı

Akış:

1. Kullanıcıdan bir not girmesi istenir.
2. Not 0-100 arasında mı diye kontrol edilir:
    - Geçerliyse harf notu belirlenir.
    - Geçerli değilse hata mesajı yazdırılır.
3. Harf notu veya hata mesajı ekrana yazdırılır.

Özet:

- Ternary if, kısa ve okunabilir kodlar yazmanızı sağlar.
- Çok katmanlı koşullarda dikkatli kullanılmalıdır; aksi takdirde kod karmaşık hale gelebilir.

Dikkat edin bu sefer kullanıcının 0 ile 100 arasında bir not aldığından bir adım daha fazla emin olduk. Siz bu kontrolü ternary if ile yapamıyor olabilirsiniz. Bunun şu an da bir önemi yok. Programınızı geniş geniş yazın ve akışı anlamaya çalışın. Gerekirse programınızı debug ile izleyin.

Gördüğünüz üzere programımızın boyutu oldukça azaldı. Ancak okuması da biraz zorlaştı. Zamanla kazandığınız tecrübe ile bu şekilde kod yazmaya başlayacaksınız. Ancak o zamana kadar en kolay okuduğunuz programı yazmaya çalışın.

SWITCH-CASE YAPISINA GİRİŞ

switch-case yapısı, birden fazla durumu kontrol etmenin alternatif ve düzenli bir yoludur. if-else if yerine tercih edilir. Özellikle sabit değerler karşılaştırılırken daha okunaklıdır.

Eski Tip Kullanım

```
switch (ifade)
{
   case deger1:
      // ifade deger1 ise bu kod çalışır.
      break;
   case deger2:
      // ifade deger2 ise bu kod çalışır.
      break;
   default:
      // Hiçbir koşul sağlanmadığında bu kod çalışır.
      break;
}
```

Örnek:

```
static void Main()
{
    Console.Write("Bir gün numarası girin (1-7): ");
    int gun = int.Parse(Console.ReadLine());

    switch (gun)
    {
        case 1:
            Console.WriteLine("Pazartesi");
            break;
        case 2:
            Console.WriteLine("Salı");
            break;
        case 3:
            Console.WriteLine("Çarşamba");
            break;
        case 4:
            Console.WriteLine("Perşembe");
            break;
        case 5:
            Console.WriteLine("Cuma");
            break;
        case 6:
            Console.WriteLine("Cumartesi");
            break;
        case 7:
            Console.WriteLine("Pazar");
            break;
        default:
            Console.WriteLine("Geçersiz bir gün numarası girdiniz.");
            break;
    }
}
```

Şekil 6 Switch-Case yapısı örnek kullanımı

Burada ki yapı kısaca şu şekildedir. Eğer elde ettiğimiz gün değeri, case içerisinde verilen sayıya eşit ise oradaki blok çalışır. Ancak diğer bloklarda kontrol edilir. Eğer hiçbir değer ile uyuşmaz ise default bloğuna girilir. Peki buradaki break ifadesi nedir? Buna daha sonra değineceğiz.

Yeni Tip Switch Kullanımı

C# 8.0 ile birlikte, daha sade ve ifade odaklı bir switch yapısı tanıtıldı. Bu yapı, özellikle değişkenlerin direkt olarak döndürüldüğü durumlarda oldukça faydalıdır.

Örnek:

```
static void Main()
{
    Console.Write("Bir gün numarası girin (1-7): ");
    int gun = int.Parse(Console.ReadLine());
    string gunAdi = gun switch
    {
        1 => "Pazartesi",
        2 => "Salı",
        3 => "Çarşamba",
        4 => "Perşembe",
        5 => "Cuma",
        6 => "Cumartesi",
        7 => "Pazar",
        _ => "Geçersiz bir gün numarası"
    };
    Console.WriteLine(gunAdi);
}
```

Şekil 7 Yeni tip switch case kullanımı

İlk sorun şu an için programımızı .net framework 4.8'de çalıştırdığımız için bu kullanımı şu an çalıştıramayız. Yeni versiyona geçtiğimizde bu ifadeyi kullanabiliriz.

Burada anlaşılmayan tek şey "_" ifadesi olabilir. Bunu default yerine kullanıyoruz. Bunu ileride daha iyi anlayacaksınız. Şimdilik bunu bilmeniz yeterlidir. Kullanıma bu şekilde gerek yoktur.

Özet:

- Eski tip switch-case yapısı daha çok prosedürel kodlar için uygundur.
- Yeni tip switch kullanımı ise daha kısa ve fonksiyonel bir yaklaşım sunar.

Şimdi gelelim "break" ifadesine. Ne anlama gelmektedir. Ne için kullanılır. Şimdi bunu inceleyelim.

AKIŞ KONTROL YAPILARI

Akış kontrolü, bir programın hangi kısmının ne zaman çalışacağını ve nerede duracağını belirler. C# dilinde kullanılan temel akış kontrol ifadeleri şunlardır:

BREAK İFADESİ

break, bir döngüyü veya bir switch-case yapısını sonlandırmak için kullanılır. Kullanıldığı yerde döngü ya da switch-case tamamlanmadan çıkış yapılır.

Örnek:

```
static void Main()
{
    for (int i = 0; i < 10; i++)
    {
        if (i == 5)
        {
            break; // Döngüden çıkılır.
        }
        Console.WriteLine(i); // 0, 1, 2, 3, 4 yazdırır.
    }
}
```

Şekil 8 Break ifadesi örnek kullanım

RETURN İFADESİ

return, metodu ya da döngüyü tamamen sonlandırır ve çağıran yere değer döndürebilir. Ancak metot kullanılmadığında bile, return kodun akışını durdurmak için kullanılabilir. Henüz metodları öğrenmediğiniz için bu konuya değinmeye gerek yoktur. Ancak metodlarda kullanım oldukça önemlidir. Metodlar konusunda dahada değinilecektir.

Örnek:

```
static void Main()
{
    for (int i = 0; i < 10; i++)
    {
        if (i == 5)
        {
            Console.WriteLine("Döngüden çıkıldı.");
            return; // Metodu sonlandırır.
        }
        Console.WriteLine(i); // 0, 1, 2, 3, 4 yazdırır.
    }

}
```

Şekil 9 Return ifadesi örnek kullanım

CONTINUE İFADESİ

continue, döngünün o anki iterasyonunu atlar ve bir sonraki iterasyona geçer.

Örnek:

```
static void Main()
{
    for (int i = 0; i < 10; i++)
    {
        if (i % 2 == 0)
        {
            continue; // Çift sayıları atla.
        }
        Console.WriteLine(i); // 1, 3, 5, 7, 9 yazdırır.
    }
}
```

Şekil 10 Continue ifadesi örnek kullanımı

GOTO İFADESİ

goto, kodun akışını belirli bir etikete (label) doğru yönlendirir. Ancak kullanımı genelde tavsiye edilmez çünkü kodun okunabilirliğini azaltabilir. Ayrıca istenmeyen sonsuz döngüye sebep olabilir.

Örnek:

```
static void Main()
{
    int sayac = 0;

baslangic:
    Console.WriteLine(sayac);
    sayac++;
    if (sayac < 5)
    {
        goto baslangic; // Etikete geri döner.
    }
}
```

Şekil 11 Goto ifadesi örnek kullanımı

THROW İFADESİ

throw, bir hata oluşturmak (exception fırlatmak) için kullanılır. Hataları kontrol etmek ve uygun mesajlar vermek için faydalıdır. Bu konuya try-catch yapısında daha fazla değinilecektir. Ancak şimdilik bilin yeter. Kullanmanıza gerek yoktur.

Örnek:

```
static void Main()
{
    int bolen = 0;
    if (bolen == 0)
    {
        throw new DivideByZeroException("Bölen sıfır olamaz.");
    }
}
```

Şekil 12 Throw ifadesi örnek kullanım

Özet

- **break**: Döngüyü veya switch-case yapısını sonlandırır.
- **return**: Metodu sonlandırır ve değer dönýr.
- **continue**: Bir döngü iterasyonunu atlar.
- **goto**: Etikete doğru akışı yönlendirir (nadiren kullanılır).
- **throw**: Hata fırlatır.

Bu ifadeler, programın akışını esnek bir şekilde yönetmenizi sağlar.

ÖDEV

Yukarıda size öğretilen kontrol yapıları ve akış kontrolünü kullanmaya çalışın. Örnek olarak bir sıcaklık ve nem sensörünü simüle edin. Bu değerleri konsoldan siz girin. Ancak size sensörlerden geldiğini var sayın. Alınan değerlere göre

- yangın var!!!!
- Yangın tehlikesi dikkatli olun!
- Buzlanma ihtimali var!

Gibi gibi uyarıları döndürün.

Ne kadar hata kontrolü yaparsanız o kadar iyi bir program yapmışsınız demektir. Lütfen programınızdaki ihtimalleri artırınız.

TEST

1. **C# dilinde break ifadesi hangi durumlarda kullanılır?**
   A) Döngüyü veya switch-case yapısını sonlandırmak için
   B) Hata oluşturmak için
   C) Etikete yönlendirmek için
   D) Bir iterasyonu atlamak için

2. **continue ifadesi döngülerde nasıl bir işlev görür?**
   A) Döngüyü tamamen sonlandırır

B) Döngünün o anki iterasyonunu atlayarak bir sonraki iterasyona geçer
C) Bir etikete yönlendirir
D) Hata fırlatır

3. **Aşağıdaki durumlardan hangisi return ifadesi ile tanımlanır?**
   A) Bir döngüyü sonlandırır
   B) Metodu sonlandırır ve çağıran yere değer döndürür
   C) Etikete yönlendirir
   D) Döngüdeki iterasyonu atlar

4. **C#'ta goto ifadesi neden genellikle tavsiye edilmez?**
   A) Kodun okunabilirliğini azaltır ve sonsuz döngülere yol açabilir
   B) Metodları sonlandırır
   C) Hata fırlatır
   D) Şartları kontrol eder

5. **Aşağıdaki ifadelerden hangisi bir hata oluşturmak için kullanılır?**
   A) continue
   B) break
   C) throw
   D) return

6. **Aşağıdaki seçeneklerden hangisi switch-case yapısında default ifadesinin işlevini açıklar?**
   A) Hiçbir koşul sağlanmadığında çalışacak kod bloğunu belirler
   B) Döngüyü sonlandırır
   C) Bir hata oluşturur
   D) Şartları kontrol eder

7. **Bir switch-case yapısında break ifadesi kullanılmazsa ne olur?**
   A) Derleyici hata verir
   B) Kod bir sonsuz döngüye girer
   C) Kod, kendisinden sonraki tüm case bloklarını çalıştırır
   D) Hiçbir şey olmaz

8. **break ve continue ifadeleri arasındaki temel fark nedir?**
   A) break döngüyü sonlandırırken, continue sadece o iterasyonu atlar
   B) continue döngüyü sonlandırır, break o iterasyonu atlar
   C) Her ikisi de aynı işlevi görür
   D) break sadece switch yapısında kullanılır

9. **C# dilinde bir döngü içindeki goto ifadesi ile sonsuz döngü oluşturulması hangi durumlarda gözlemlenir?**
   A) goto etiketi yanlış tanımlandığında
   B) Döngü doğru şekilde sonlandırılmadığında
   C) goto bir etiketi tekrar tekrar çağırdığında
   D) goto yalnızca switch içinde kullanıldığında

10. **Yeni tip switch yapısında _ (alt çizgi) sembolünün işlevi nedir?**
    A) Hiçbir koşul sağlanmadığında çalışacak varsayılan bloğu temsil eder
    B) Döngüyü sonlandırır

C) Değer döndürmek için kullanılır
D) Etikete yönlendirme yapar

CEVAP ANAHTARI

1. A
2. B
3. B
4. A
5. C
6. A
7. C
8. A
9. C
10. A