Swift Programlama Dili ile iOS Mobil Uygulama Geliştirme

MURAT ADIGÜZEL (murat.adiguzel@intertech.com.tr)

5 Nisan 2021

Temel Kontrol Akışı

Egzersizler (Döngüler)

1. `counter` adlı bir değişken oluşturun ve `0`'a eşit olarak ayarlayın. `counter` < `10` koşulu ile bir `while` döngüsü oluşturun. Döngü içinde `counter` değişkeninin o anki durumunu yazırın (`"Counter is X"`) ve ardından `counter` değişkenini `1` artırın.

2. `counter` adında bir değişken oluşturun ve `0`'a eşitleyin. `roll` adında başka bir değişken oluşturun ve onu da `0`'a eşitleyin. Bir `repeat-while` döngüsü oluşturun. Döngünün içinde `roll` değişkenini `Int.random(in: 0...5)`'e eşit olacak şekilde ayarlayın. Bu, `0` ile `5` arasında rastgele bir sayı seçmek anlamına gelir. Ardından `counter`'ı `1` artırın. Döngünün içinde son olarak, `"After X rolls, roll is Y"` mesajını yazdırın. Burada `X` `counter` değeri, `Y` ise `roll` değeridir. Döngü koşulunu, `roll` ilk olarak `0` değerini aldığında bitecek şekilde ayarlayın.

Temel Kontrol Akışı

Egzersizler

1. Koddaki hata nedir?

```
let firstName = "Murat"
if firstName == "Murat" {
  let lastName = "Adıgüzel"
} else if firstName == "Deniz" {
  let lastName = "Yıldız"
}
let fullName = firstName + " " + lastName
```

2. Aşağıdaki ifadelerin her birinde, `Bool` türündeki **`answer`** sabitinin değeri nedir?

```
let answer = true && true
let answer = false || false
let answer = (true && 1 != 2) || (4 > 3 && 100 < 1)
let answer = ((10 / 2) > 3) && ((10 % 2) == 0)
```

Temel Kontrol Akışı

Egzersizler

3. 1. pozisyondan 20. pozisyona giden bir yılan ve merdiven oyunu oynadığınızı hayal edin. Üzerinde, 3. ve 7. pozisyonlarda sizi sırasıyla 15 ve 12'ye götüren merdivenler var. Ayrıca 11. ve 17. pozisyonlarda sizi sırasıyla 2 ve 9'a götüren yılanlar var.

   1 ile 20 arasında istediğiniz herhangi bir konuma ayarlayabileceğiniz `currentPosition` adlı bir sabit oluşturun. Ardından, istediğiniz zar atışına ayarlayabileceğiniz, `diceRoll` adlı bir sabit oluşturun. Daha sonra, merdivenleri ve yılanları hesaba katarak son konumu hesaplayın ve bunu `nextPosition` olarak adlandırın. Son olarak da tahta üzerinde bulunduğunuz son konumu `"Board position after X is Y"` şeklinde yazdırın. Burada **X** ilk pozisyonu, **Y** ise son pozisyonu temsil ediyor.

Temel Kontrol Akışı

Egzersizler

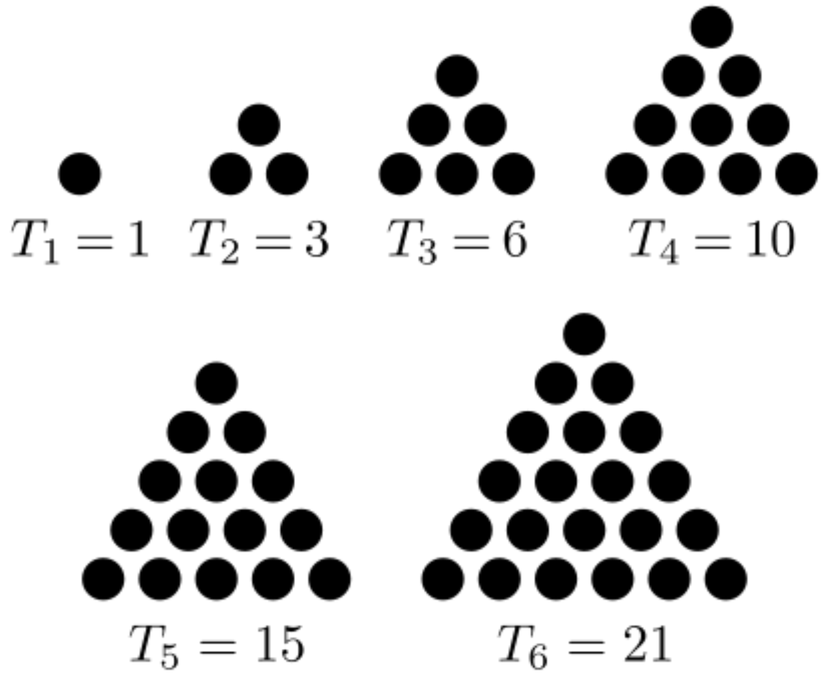
4. Bir ay (tümü küçük, 3 harfli bir `String` ile ifade edeceğiz) ve geçerli yıl (`Int` ile ifade edeceğiz) verildiğinde, aydaki gün sayısını hesaplayın. Artık yıllar nedeniyle, "Şubat"'ın yıl, 4'ün katı olduğu ancak 100'ün katı olmadığı zamanlarda 29 gün olduğunu unutmayın. Bunun yanında şubat ayı, yılın 400'ün katı olduğu zamanlarda da 29 güne sahiptir.

5. Verilen sayıdan büyük veya eşit olan 2’nin en küçük kuvvetini bulacak kodu yazın.

Temel Kontrol Akışı

Egzersizler

6. Bir sayı verildiğinde, o derinliğe sahip üçgen sayısını yazdırın.



Temel Kontrol Akışı

Egzersizler

7. n'inci Fibonacci sayısını hesaplayın. Fibonacci dizisinde ilk iki sayının 1 ile başladığını ve ardından dizideki sonraki sayıların ondan önceki iki sayının toplamına eşit olduğunu unutmayın.

```
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …
```

Temel Kontrol Akışı

Egzersizler

8. Verilen sayının 12'ye kadar olan çarpım tablosunu döngü kullanarak yazdırın.

Temel Kontrol Akışı

Özet

1. Doğru ve yanlışı temsil etmek için `Bool` veri türünü kullanırsınız.

2. Hepsi bir Boolean döndüren karşılaştırma operatörleri şunlardır:

Name	Operator
Equal	==
Not Equal	!=
Less than	<
Greater than	>
Less than or equal	<=
Greater than or equal	>=

Temel Kontrol Akışı

Özet

3. Karşılaştırma koşullarını birleştirmek için Boolean mantığını **(&& ve ||)** kullanabilirsiniz.

4. Bir koşula göre basit kararlar vermek için **if** ifadelerini kullanırsınız.

5. Karar verme sürecini tek bir koşulun ötesine genişletmek için bir **if** ifadesinde **else if** ve **else** kullanırsınız.

6. Bir Boolean ifadesinin yalnızca minimum gerekli kısımları değerlendirilir.

7. Basit **if** ifadeleri yerine ternary operator **(a ? b : c)** kullanabilirsiniz.

8. Değişkenleri ve sabitleri yalnızca tanımlandığı kapsam içinde kullanabilirsiniz. Bir kapsam, üst kapsamdaki değişkenleri ve sabitleri kullanabilir.

Temel Kontrol Akışı

Özet

9. `while` döngüleri, bir koşul karşılanana kadar belirli bir görevi birkaç kez gerçekleştirmenize izin verir.

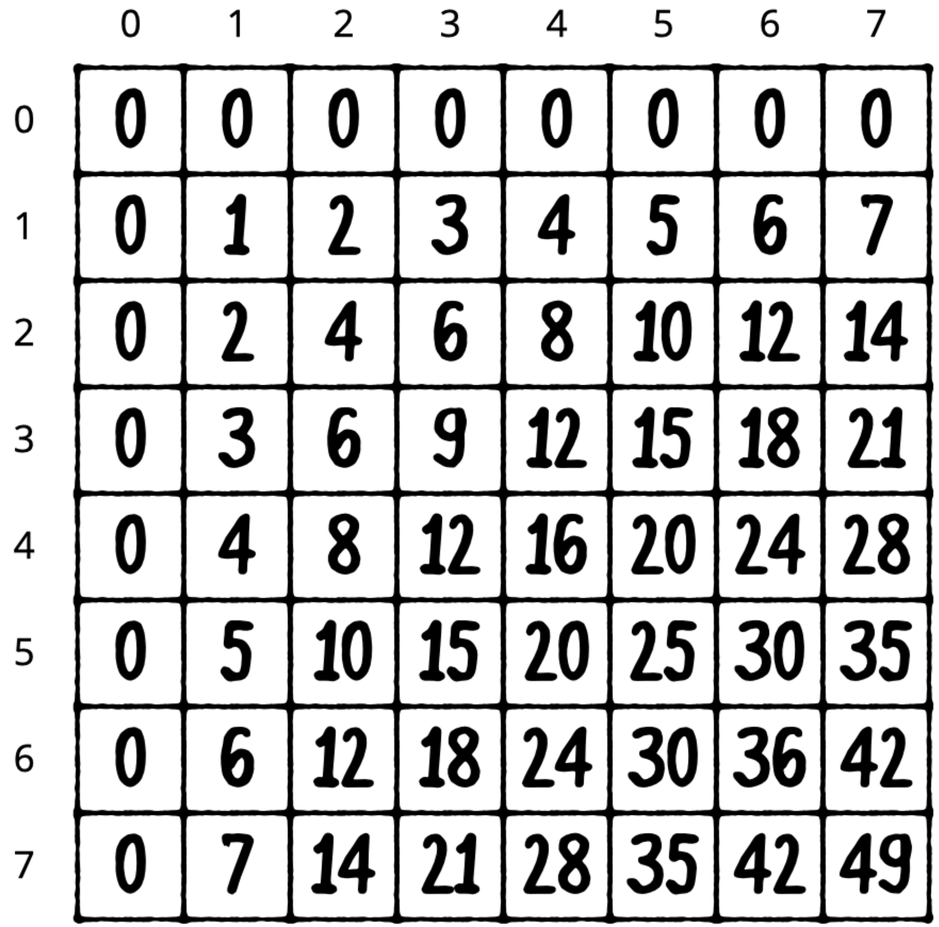
10. `repeat-while` döngüleri her zaman döngüyü en az bir kez çalıştırır.

11. `break` anahtar kelimesi, bir döngüden çıkmanıza izin verir.

Gelişmiş Kontrol Akışı

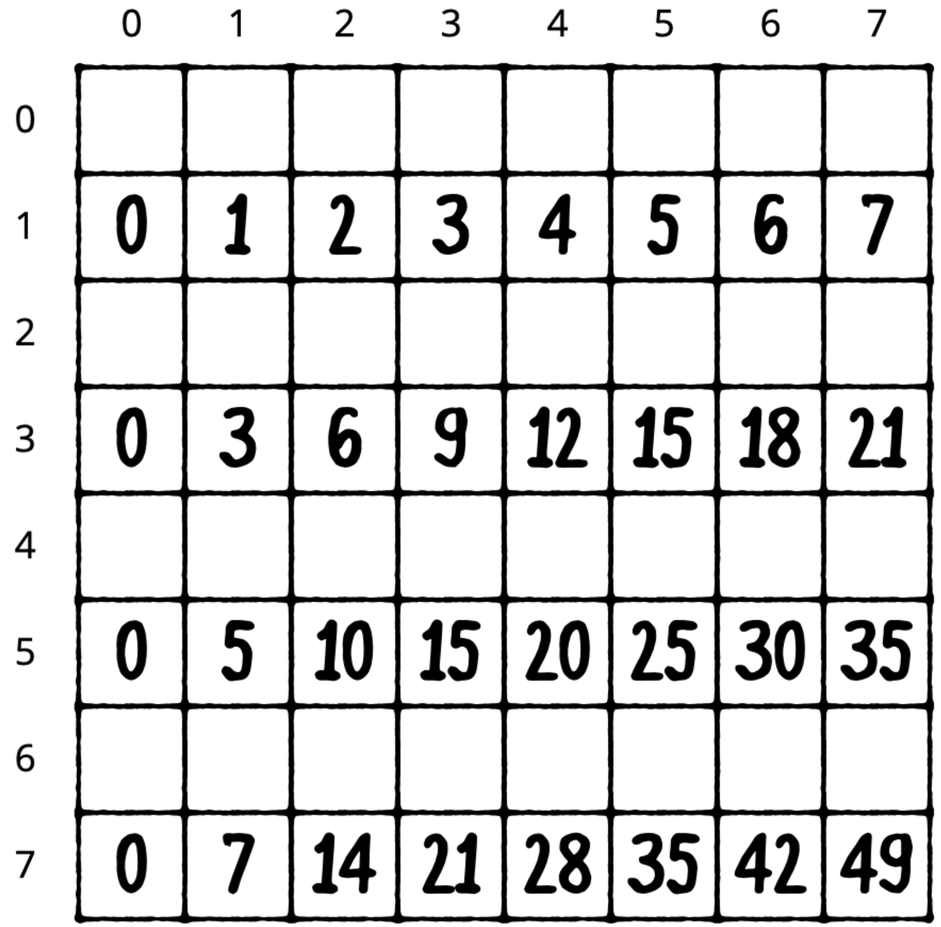
Gelişmiş Kontrol Akışı

Continue statement



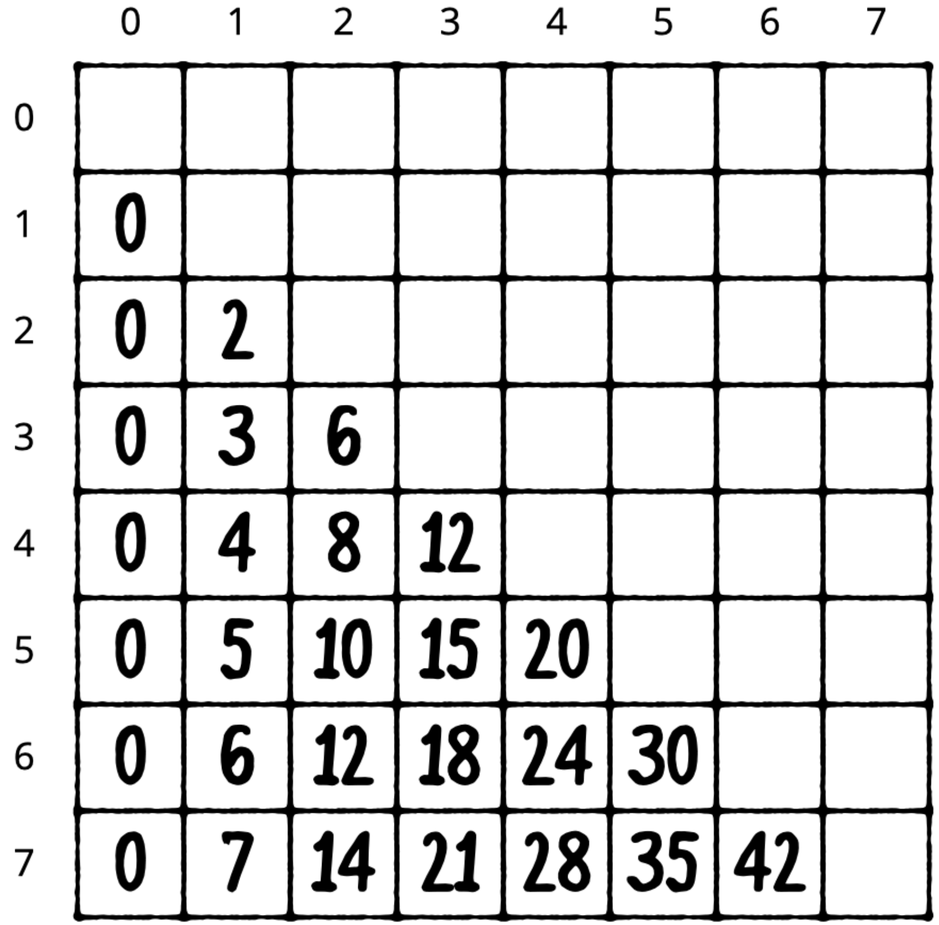
Gelişmiş Kontrol Akışı

Continue statement



Gelişmiş Kontrol Akışı

Continue statement



Gelişmiş Kontrol Akışı

Egzersizler (for döngüleri)

1. `range` adında bir sabit oluşturun ve bunu `1`'den başlayıp `10` dahil olmak üzere biten bir aralığa eşit olarak ayarlayın. Bu aralığı kullanarak, aralıktaki her bir sayının karesini konsola yazdıran bir `for` döngüsü yazın.

2. Yukarıdaki alıştırmada olduğu gibi aynı aralık üzerinde bu kez her sayının karekökünü yazdıran bir `for` döngüsü yazın.

Gelişmiş Kontrol Akışı

Egzersizler (for döngüleri)

3. **8x8** ızgarada yalnızca tek satırları yazdıran kod bloğu görmüştünüz.

```
var sum = 0
for row in 0..<8 {
  if row % 2 == 0 {
    continue
  }
  for column in 0..<8 {
    sum += row * column
  }
}
```

Devam etmek yerine çift satırları atlamak için ilk **`for`** döngüsünde bir **`where`** cümlesi kullanacak şekilde değişiklik yapın. İlk örnekte olduğu gibi toplamın **448** olduğunu kontrol edin.

Gelişmiş Kontrol Akışı

Egzersizler (switch ifadeleri)

1. Yaşı tam sayı olarak alan ve o yaşla ilgili yaşam evresini yazdıran bir **`switch`** ifadesi yazın. Sınıflandırmayı şu şekilde düşünebilirsiniz: **`0–2 yaş Bebek`**; **`3–12 yaş Çocuk`**; **`13–19 yaş Genç`**; **`20–39 yaş Yetişkin`**; **`40–60 yaş Orta yaşlı`**; **`61+ yaş Yaşlı`**.

2. Bir **`String`** ve bir **`Int`** içeren bir tuple kontrol eden bir **`switch`** ifadesi yazın. **`String`** bir isimdir ve **`Int`** bir yaştır. Önceki alıştırmada kullandığınız durumların aynısını kullanın ve önce adı ve ardından yaşam evresini yazdıran bir ifade yazın. Örneğin, benim için **`"Murat bir yetişkin"`** yazdırması gerekir.

Gelişmiş Kontrol Akışı

Egzersizler

1. Aşağıdaki **`for`** döngüsünde, toplamın değeri ne olacak ve kaç tane yineleme olacak?

```
var sum = 0

for i in 0...5 {

  sum += i

}
```

Gelişmiş Kontrol Akışı

Egzersizler

2. Aşağıdaki **`while`** döngüsünde, **`aLotOfAs`** değişkeninde kaç tane **“a”** harfi olacak? İpucu: **`aLotOfAs.count`** size **`aLotOfAs`** string’i içinde kaç karakter olduğunu söyler.

```
var aLotOfAs = ""

while aLotOfAs.count < 10 {

  aLotOfAs += "a"

}
```

Gelişmiş Kontrol Akışı

Egzersizler

3. Aşağıdaki switch ifadesini düşünün. Koordinatlar sağdakilerden her biri olduğunda bu kod ne yazdırır?

```
switch coordinates {
case let (x, y, z) where x == y && y == z:
  print("x = y = z")
case (_, _, 0):
  print("On the x/y plane")
case (_, 0, _):
  print("On the x/z plane")
case (0, _, _):
  print("On the y/z plane")
default:
  print("Nothing special")
}
```

```
let coordinates = (1, 5, 0)
let coordinates = (2, 2, 2)
let coordinates = (3, 0, 1)
let coordinates = (3, 2, 5)
let coordinates = (0, 2, 4)
```

Gelişmiş Kontrol Akışı

Egzersizler

4. Kapalı bir aralık asla boş olamaz. Neden?

5. **`10`**'dan **`0`**'a geri sayım yazdırın.

6. **`0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0`** yazdırın.

Gelişmiş Kontrol Akışı

Özet

1. Sayılabilir aralıkları (Countable ranges), bir değerden diğerine geçen artan bir tamsayı dizisi oluşturmak için kullanabilirsiniz.

2. Kapalı aralıklar (Closed ranges) hem başlangıç hem de bitiş değerlerini içerir.

3. Yarı açık aralıklar (Half-open ranges), başlangıç değerini içerir ve bitiş değerinden bir önce durur.

4. `for` döngüleri, bir aralıkta yineleme yapmanıza izin verir.

5. `continue` ifadesi, bir döngünün geçerli yinelemesini bitirmenize ve bir sonraki yinelemeye başlamanıza olanak tanır.

Gelişmiş Kontrol Akışı

Özet

6. Etiketli ifadeler dış döngüde `continue` ve `break` kullanmanıza izin verir.

7. Bir değişkenin veya sabitin değerine bağlı olarak hangi kodun çalıştırılacağına karar vermek için `switch` ifadesini kullanırsınız.

8. Bir `switch` ifadesinde, karmaşık kurallar kullanarak değerleri karşılaştırmak için desen eşleştirmeden (pattern matching) yararlanırsınız.