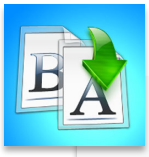
KARAR YAPILARI VE DÖNGÜLER



İÇİNDEKİLER

- Karar Yapıları
  - If - else Yapısı
  - Switch- case Yapısı
- Döngüler
  - For Döngüsü
  - While Döngüsü
  - Do-while Döngüsü
  - Foreach Döngüsü

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Karar mekanizmalarının çalışma mantığını anlayacak,
  - Döngü yapılarının nasıl gerçekleştiğini anlayacak,
  - Karar mekanizmalarını ve döngü yapılarını beraber kullanabilmeyi öğreneceksiniz.

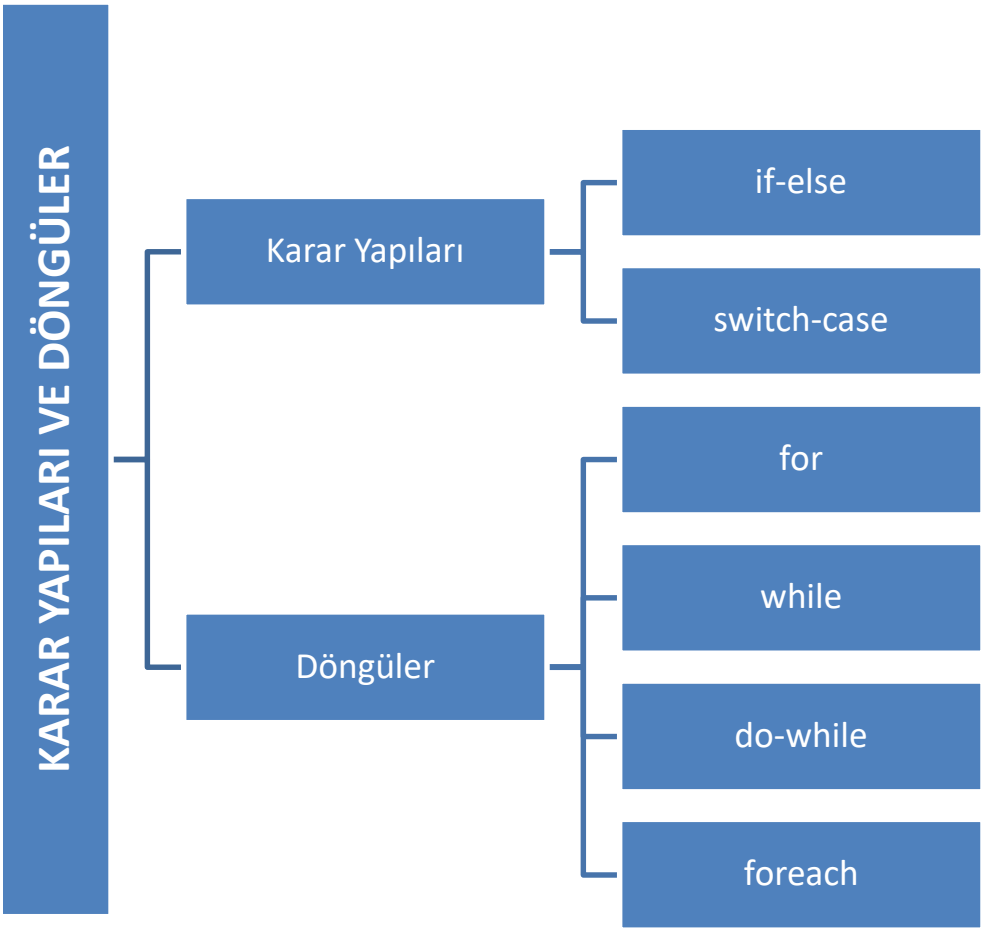
Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Öğr. Gör.
Çağrı Burak ARINÇ

ÜNİTE

5


KARAR YAPILARI VE DÖNGÜLER
Karar Yapıları
if-else
switch-case
Döngüler
for
while
do-while
foreach

GİRİŞ

Programlama dillerinde sistematik bir akış bulunmaktadır. Sistematik akış, temel olarak tasarlanan kodun baştan sona doğru ilerlemesi ile gerçekleşir. Söz konusu akış içerisinde karar verme süreçlerinin kullanılması gereken durumlar bulunmaktadır. Karar verme mekanizması sayesinde sistematik akış farklılaşarak farklı kod bölümlerde işlem yapabilme kabiliyetine sahip olmaktadır. Ayrıca döngüler sayesinde belirlenen kod blokları tekrar tekrar çalıştırılarak, sistematik tekrarlanan akış tasarlanabilmektedir. Bazı durumlarda karar verme mekanizmaları yerine daha sade yapılar da kullanılabilmektedir.

Programlama dillerinin temel yapılarından olan karar verme mekanizmaları ve döngüler hemen hemen tüm dillerde aynı yapıya sahiptir. Bu ünitede c# dili temel alınacaktır. Karar mekanizmaları için "if-else" ve "switch- case" yapıları döngüler için de "for", "while", "do-while" ve "forerach" incelenerek akış mekanizmalarının nasıl gerçekleştiği açıklanacaktır.

KARAR YAPILARI

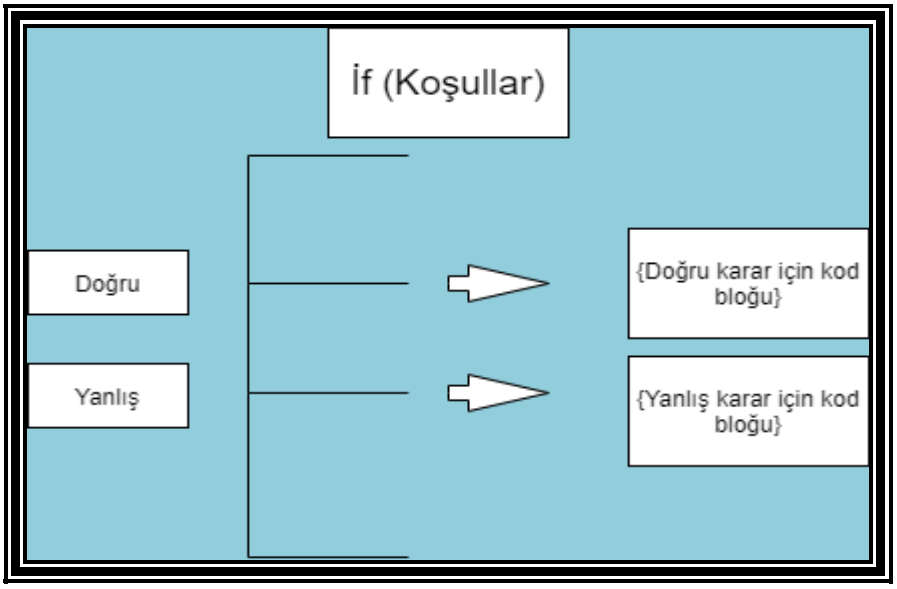
Karar yapıları bilgisayarın iki ya da daha fazla akış bulunması durumunda seçim yapabilmesini sağlayan mantık yapılarıdır. Karar yapıları olmadan yapılacak işlemler, basit hesaplamaların ötesine geçemeyecektir. Karar mekanizması günlük yaşamımızda yaptığımız her seçime benzer özellikler sağladığından anlaşılması kolay mantıksal gruplardandır. Günlük yaşantımız ile bu mantıksal yapıyı açıklamak istersek; hayatımızda yaptığımız seçimler bu mekanizma için doğru bir tanımlama olacaktır. Örneğin bilgisayar satın almak istediğimizde kendimize özellik listesi yapar ve bu doğrultuda inceleme yapmaya başlarız, karar mekanizması bu aşamada karşımıza çıkar ve şu sorunun cevabını arar? Eğer ürün listemdeki özelliklere sahipse alabilirim, eğer sahip değilse başka bir ürün incelemeliyim.

If ifadesi "eğer" anlamına gelmektedir.

C# programlama dilinde karar yapıları incelendiği zaman, karşımıza iki farklı yapı çıkmaktadır. Bunlar: "if"(eğer) – "else" (başka) ve "switch" (değişim) - "case"(durum)'dir. Bu yapılar kullanılarak tek koşullu, çok koşullu ve iç içe karar yapıları oluşturulabilir.

If- Else Yapısı

Programlamanın temelinde yazılan kodların belli bir akış içerisinde ilerlediğinden bahsetmiştik. Bu akışın farklılaştırılabilmesi için gerekli olan yapı "if" (eğer) ve "else" (başka) ile *gerçekleşmektedir*. "If" ve "else" kod bloğu verinin doğru ya da yanlış olmasını baz alarak akışı değiştirir ve bazı kod bloklarının çalıştırılmamasını sağlar. Else kullanımı zorunlu değildir, else kullanılmadığı zaman sadece "if" içerisindeki koşulun sağlanması şartı aranarak işlem yapılabilir veya çıktı alınabilir.



Şekil 5.1. İf- Else Yapısı

Şekil 5.1'de görüldüğü gibi "if" (eğer) ile belirttiğimiz koşulların doğru olması durumunda ilk kod bloğuna girecek ve yanlış olan kod bloğu çalışmayacaktır. "If" ile belirtilen koşulların yanlış olması durumunda ise sadece yanlış kararlar için yazılmış olan kod bloğu çalışacaktır.

Aşağıda verilen örnekte tek koşullu bir yapı tasarlanmıştır. Saatin 12'den küçük olması "if" ile belirtilen koşul ifadesi ile kontrol edilmiştir. Koşulun 12'den büyük olması sonucu kod bloğu yanlış olan yani "else" bölümüne girerek işlemi gerçekleştirmiş ve çıktı olarak öğleden sonra yazısını vermiştir. Eğer saat değişkeni 14 olarak değil 12'den ufak bir sayı olarak tanımlanmış olsaydı kontrol mekanizması koşulu doğru olarak kabul ederek kod bloğunu "if" (doğru) bölümünde değerlendirmeye alarak çıktıyı farklılaştıracaktı.

Örnek

```
int saat= 14;
if (saat< 12)
{
  MessageBox.Show("Öğleden Önce.");
}
else
{
  MessageBox.Show("Öğleden Sonra.");
}
```

Çok koşullu karar mekanizmalarını tasarlamak için üç farklı yapı bulunmaktadır. Bunlar düz, pozitif ve negatif mantık olarak adlandırılır. Düz mantıkta verilen tüm koşullar doğrusal olarak işlenir, yani "else" kod bloğu ile ilgili bir işlem yapmaz ve sadece sonucun yanlış olduğu durumlarda yeni koşul ifadesine

geçer. Pozitif mantıkta ise “if” (eğer) ile belirtilen koşul ifadeleri doğru ise, doğru için yazılan kod bloklarını çalıştırmak yerine programın akışı devam ettirilir. Negatif mantıkta ise program akışı, koşul ifadesi yanlış değer döndürdüğü sürece oluşur.

Karar mekanizmaları tasarlanırken birden fazla koşulun aynı anda gerçekleşmesi durumu aranabilir. Birden fazla koşulun aynı anda sağlanabilmesi için iç içe geçmiş koşul ifadelerinin yazılması gerekebilir. Aşağıda gösterilen örnekte “a” ve “b” değişkeni tanımlanmış, iç içe iki koşul içerinde bu değerlerin doğru olduğu kontrol edilerek çıktının alınması sağlanmıştır.

Örnek

```
int a = 1;
 int b = 2;
      if (a == 1) {
        if (b == 2) {
          MessageBox.Show("ikici if bloğu");
                    }
                   }
```

“Else if” yapısında sadece “if” koşulu “false”(yanlış) olarak değerlendirildiğinde akış devam ettirilir. Bu nedenle, “if” veya “else if” ifadelerinden sadece biri akışı devam ettirebilir, ancak ikisi birden çalıştırılamaz. Aşağıda verilen örnekte i ve j değişkeni tanımlanmış ve “else if” karar mekanizması ile sayıların eşit olup olmadığı veya birbirlerinden büyük ya da küçük olup olmadığı kontrol edilmiştir.

Örnek

```
int i = 10, j = 20;
if (i == j)
{ MessageBox.Show("i ve j değeri eşittir"); }
 else if (i > j)
 { MessageBox.Show("i değeri j değerinden büyüktür"); }
 else if (i < j)
 { MessageBox.Show("i değerii j değerinden küçüktür"); }
```

Bireysel Etkinlik

- Hastahanede çalışan bir doktorun hastaları için yazacağı ilacın dozajını "if - else" karar mekanizması ile belirleyen bir kod bloğu yazmaya çalışınız. Kullanıcıdan istenen hasta yaşına bağlı olarak:
- 0-6 yaş 10mg
- 7-18 yaş 20 mg
- 18+ yaş 25 mg olarak akışı tanımlayınız.

"Break" ifadesi her "case" ifadesinden sonra kullanılmalıdır.

Switch - Case Yapısı

Bu yapı farklı durumların dallanması ile oluşan ifadelerde "if-else" blokları yerine kullanılabilen karar mekanizmasıdır. "Switch –case" yapısı ile tasarlanan kod blokları "if-else" yapısı ile de yapılabilecek olsa da daha sade ve karmaşıklığı önleyebilecek bir yapıya sahip olması nedeniyle kullanılmaktadır. "Switch-case" yapısının çalışma mantığı incelediğinde öncelikle "switch" ifadesi içerisinde bulunan parantez işleme alınır ve içerisinde bulunan ifadenin değeri hesaplanır. Hesaplanan değerle eşleşen "case" ifadesi bulunursa, o bloktaki akış çalıştırılır. Eğer hiçbir "case" bloğu ile "switch" ifadesindeki koşul değeri eşleşme sağlamazsa default bloğundaki kodlar çalıştırılır. "Break" ifadesi, her "case" bloğundan sonra mutlaka kullanılmalıdır. Çünkü istenen kod bloğu çalıştırılmış olmasına rağmen, "break" ifadesi kullanılmazsa "switch" dışına çıkılmadan aşağıdaki "case" bloklarına doğru akış devam eder.

Örnek

```
string mevsim = "yaz";
    switch (mevsim)
      {
      case "kış":         MessageBox.Show("Aralık, Ocak, Şubat"); break;
      case "ilkbahar": MessageBox.Show("Mart, Nisan, Mayıs"); break;
      case "yaz":         MessageBox.Show("Haziran, Temmuz, Ağustos");
 break;
      case "sonbahar": MessageBox.Show("Eylül, Ekim, Kasım"); break;
      default: MessageBox.Show("Hatalı veri");  break;
        }
```

Yukarıda verilen örnekte "switch –case" yapısı oluşturularak "mevsim" değişkeninde belirtilen mevsimin hangi aylardan oluştuğunu bulan bir kod bloğu tasarlanmıştır. Mevsim değeri "yaz" olarak tanımlanmıştır. Sadece üçüncü "case" değeri "true"(doğru) sonuç verdiği için sadece bu kod bloğu akışı devam ettirmiştir. Farklı bir değerlendirme ele alırsak mevsim değişkenini kullanıcıdan almış olsaydık ve hiçbir "case" içerinde bulunan değer ile eşleşmeseydi "default" kod bloğu akışa devam ederek hatalı veri çıktısı verecekti.

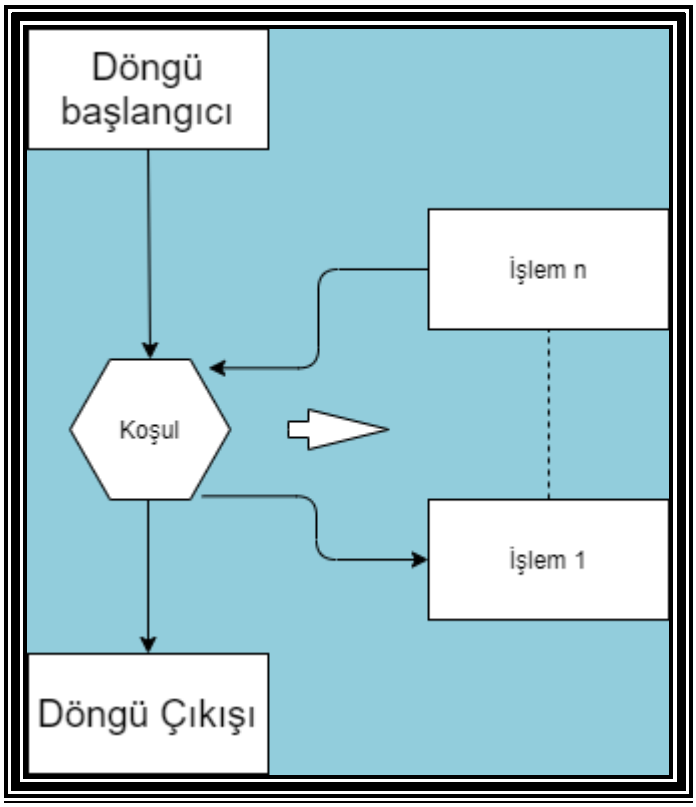
Bireysel Etkinlik

- Trafik lamba renklerine göre sürücünün nasıl davranması gerektiğini belirleyen C# kod bloğunu Switch-case yapısı ile yazın. (Kırmızı : Dur, Sarı : Hazır ol, Yeşil: İlerle)
- Yazdığınız kod bloğunu "if - else" yapısı ile düzenlemeye çalışınız.

DÖNGÜLER

Programlama dillerinde tasarlanan akış yapısı birçok kez çalıştırılabilir. Tekrarlanan akışı çalıştırabilmek için kod bloğu bir kez yazılır ve döngü ifadeleri ile tekrar tekrar çalıştırılır. Programlama dillerinin temel yapılarından olan döngüler neredeyse her dilde aynı yapıya sahiptir. C# programla dili incelendiği zaman karşımıza dört farklı döngü yapısı çıkmaktadır. Söz konusu dilde belirtilen yapılar “for”, “while”, “do-while” ve “foreach” olarak tanımlanmıştır.

Döngüler sayesinde aynı kod bloğunu tekrar tekrar kullanabilirsiniz.



Şekil 5.2. Döngü Yapısı

Şekil 5.2’ de görüldüğü gibi döngü içerisinde bulunan koşul ifadesi doğru olduğu sürece işlem “n” sayısınca tekrarlanacaktır.

For Döngüsü

C# dilinde bulunan “for” döngüsü bir kod bloğunun birden fazla akış yapabilmesini sağlayan ifadedir. “For “döngüsü sayesinde birden fazla iç içe döngü

de kurulabilmektedir. “For” döngüsü, belirtilen koşul ifadesi “false”(yanlış) değer döndürene kadar art arda ifadeyi yürütür. Aşağıda verilen örnekte “for” döngüsünün yapısı gösterilmiştir.

Örnek

```
•for (initializer (başlatıcı); condition (şart); iterator (yineleyici))
•{
•    //döngüye ile çalıştırılacak kod bloğu
•}
```

For döngüsünde belirtilen koşul ifadesi “true” (doğru) olduğu sürece döngü devam eder.

“For” döngüsü, noktalı virgülle ayrılmış üç isteğe bağlı bölümü içerir:

- Başlatıcı: Bu bölüm, “for” döngüsünde değişkeni başlatmak için kullanılır. Ayrıca sıfır veya daha fazla atama ifadesi, yöntem çağırma, artırma veya eksiltme ifadesi olabilir.
- Koşul: Doğru veya yanlış döndürecek bir “bool” ifadesidir. Bir ifade doğru olarak değerlendirilirse döngüyü yeniden yürütür aksi takdirde döngüden çıkılır.
- Yineleyici: Döngü değişkeninin artışını veya azalmasını tanımlar.

Örnek

```
•    for (int i = 0; i < 3; i++)
•    {
•    MessageBox.Show("i sayısının değeri: " + (i));
•    }

• // çıktı:
• // i sayısının değeri: 0
  //  i sayısının değeri: 1
  //  i sayısının değeri: 2
```

Yukarıda verilen örnekte “For” döngüsü sayesinde “i” sayısının değeri üç defa tekrarlanarak yazdırılmıştır. “For” döngüsü incelendiğinde başlatıcı “0” olarak belirlenmiştir. Koşul ifadesi “i” değerinin 3’den küçük olduğu sürece işlemi devam ettirmesi anlamına gelmektedir. Yineleyici ifadesi ise her işlem sonrası “i” değerini 1 artırmak anlamına gelmektedir.

Bireysel Etkinlik

- 0' dan 10' a kadar olan sayıların toplamını bulan kod bloğunu "for" döngüsü kullanarak yazınız.

While Döngüsü

C#, belirtilen koşul ifadesi “false” (yanlış) değer döndürdüğü sürece kod bloğunu tekrar tekrar yürütmek için “while” döngüsünü kullanır. “For” döngüsünde olduğu gibi “while” döngüsünde de birden fazla döngü iç içe kullanılabilir. Aşağıdaki örnekte “while” döngüsünün yapısı gösterilmiştir.

Örnek

```
while (condition (Koşul))
{
   // kod bloğu
}
```

“While” döngüsü yapısında, başlatıcı (initialization) döngü öncesinde kullanılmalı, artırma ve azaltma adımları döngü içinde tasarlanmalıdır. Aşağıda verilen örnekte döngü dışarısında “i” değişkeni “0” olarak tanımlanmıştır. Koşul ifadesi “i” değerinin 10’dan küçük olduğu sürece işlemi devam ettirmesi anlamına gelmektedir. Yineleyici ifadesi “for” ifadesinden farklı olarak döngü içerisinde düzenlenmiş ve “i” değeri her döngüde 1 artırılmıştır.

Örnek

```
int i = 0; // initialization (başlatıcı)
while (i < 10) // condition (şart)
{
MessageBox.Show("i = " + (i));
 i++; // increment (yineleyici)
}
```

Bireysel Etkinlik

- 4' den 17' ye kadar olan sayıların toplamını bulan kod bloğunu "while" döngüsü kullanarak yazınız.

Döngüde { } sembolü blok işlevini görür.

Do-While Döngüsü

C# programlama dilinde kullanılan “do-while” döngüsünün yapısı, “while” ile benzerlik göstermektedir. “Do- while” döngüsü, koşulun doğruluğunu kontrol

etmeden en az bir kere çalıştırılması durumunda kullanılan bir yapıdır. Bu durumun oluşmasının sebebi program akışının ilerle biçimine dayanmaktadır. “While” döngüsünde koşul ifadeleri en başta kontrol edilirken; “do-while” döngüsünde işlem yapıldıktan sonra döngü şartları kontrol edilir bu nedenle koşul ifadesi ne sonuç verirse versin işlem sonucu çıktı olarak alınır. Aşağıdaki örnekte “do-while” yapısı gösterilmiştir.

Örnek

```
do
{
    //kod bloğu
}
while(condition(Koşul));
```

Aşağıda verilen örnekte “do- while” döngü yapısı kullanılarak 0’dan başlayıp 5 e kadar olan sayıların ekrana yazdırılması sağlanmıştır.

Örnek

```
int i = 0;
do
{
MessageBox.Show("i = "+  (i));
i++;
} while (i < 5);
```

Aşağıda verilen örnekte koşul ifadesi “false” (yanlış) değer döndürmesine rağmen “do” içerisindeki kod bloğu bir kez çalışarak “5” değerini çıktı olarak vermiştir.

Örnek

```
int i = 5;
do
{
MessageBox.Show ((i.ToString()));
i++;
} while (i<4);
```

Foreach Döngüsü

Bu döngü, dizi (array) ve koleksiyon (collection) tabanlı nesnelerin elemanlarına erişip bu elemanlar üzerinden işlem yapılabilmesini sağlar. Aşağıdaki örnekte "foreach" yapısı gösterilmiştir.

Örnek

```
foreach( değişkenTürü değişkenAdı in diziAdı )
{
  //Dizinin her elemanı için yapılacak işlemler kod bloğu
}
```

"Foreach" döngüsü ifadesi üç bölüm içerir;

- Değişken Türü: Döngüde kullanılacak ve içinde farklı değerlerin bulunacağı değişkenin "string", "int", "byte" gibi ilgili tipinin belirtildiği alan.
- Değişken Adı: Döngü için kullanılacak değerdir.
- DiziAdı: Dizi (array) gibi liste yapılarıdır.

Örnek

```
int[] sayilar = { 1, 2, 3};
int carpim = 1;
 foreach (int x in sayilar)
  {
   carpim = carpim*x;
  }
```

Yukarıda verilen örnekte üç elemanlı bir dizinin içindeki verilerin birbirleri ile çarpım sonucunu bulan bir kod bloğu tasarlanmıştır. Döngü dışarısında "sayilar" isimli dizi ve sonucu saklayabilmek için "carpim" isimli bir değişken tanımlanmış, dizi içerindeki her bir değer "foreach" döngüsü sayesinde tek tek işleme alınarak "carpim" isimli değişken ile işleme alınmıştır.

Bireysel Etkinlik

- 5 Adet sayısal değeri olan bir dizi tanımlayınız.
- Tanımladığınız dizi içerisindeki her bir değeri toplayan "foreach" döngüsünü tasarlayınız.

Sonsuz Döngü

C# programla dilinde kullanılan döngü yapılarında verilen koşulun hiçbir zaman yanlış değer vermediği durumda akış sonsuz duruma geçer. Bu döngüler, ancak "break" komutu gibi harici bir durdurma mekanizması ile sonlandırılabilir. Aşağıdaki örnekte sonsuz döngü örnekleri verilmiştir.

Örnek

```
•for( ; ; )
 {
 MessageBox.Show("Bu bir sonsuz döngüdür!");
 }
•//veya
•while(true)
 {
 MessageBox.Show("Bu bir sonsuz döngüdür!");
 }
 //veya
•do
 {
 MessageBox.Show("Bu bir sonsuz döngüdür!");
 }while(true)
```

Bireysel Etkinlik

- Sonsuz bir döngü tasarlayarak çıktısını gözlemleyiniz.
- Tasarladığınız sonsuz döngüyü sonlandırmak için "break" ifadesini kullanmaya çalışınız.

Aşağıda verilen örnekte birçok döngü yapısının beraber kullanıldığı bir kod bloğu verilmiştir. Her bir döngünün çalışma mantığını incelersek;

- döngü yani "while" döngüsünde sayaç değişkeni döngüyü iki defa çalıştırarak döngüde "sayaç değişkenini aldığı değeri 0 ve 1" olarak yazdırmıştır.
- döngü "do-while" incelendiğinde "sayi" değeri birden başladığı için döngü sadece bir defa çalışarak "sayı değişkeninin aldığı değer 1" olarak çıktı vermiştir.
- döngü ise bir "for" döngüsüdür. "For" döngüsünde bulunan başlatıcı ve şart bölümleri incelendiğinde başlatıcı yani "index" değişkeni 1 den başlamış ve şart bölümü bu değerin 10'dan küçük olduğu sürece döngünün tekrar etmesi gerektiğini belirtmiştir. Bu nedenle döngü 9 defa tekrar yaparak "döngüde index değişkeninin aldığı değer 1--9" olarak yazdırılmıştır.

- döngüde “for” döngüsü kullanılarak “char” değerlerin de “for” içerisinde kullanılabildiğini göstermek amaçlanmıştır.
- döngüde iç içe “for” döngüsünün kullanımı gösteren bir kod bloğu tasarlanmıştır. En içte bulanan ve üç kez tekrar eden “for” döngüsünü bu döngünün dışında bulunan ”for” döngüsü ile iki kez daha tekrar yaptırarak altı adet farklı çıktı alınması sağlanmıştır.

Örnek

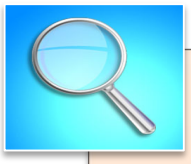
```
int sayac = 0;
while (sayac < 2)
{
MessageBox.Show("Döngüde sayac değişkeninin aldığı değer " + (sayac));
sayac++;
}

int sayi = 1;
do
 {
 MessageBox.Show("Döngüde sayi değişkeninin aldığı değer " + (sayi));
 sayi++;
 } while (sayi < 2);

 for (int index = 1; index < 10; index++)
 {
 MessageBox.Show("Döngüde index değişkeninin aldığı değer " + (index));
 }

 for (char column = 'a'; column < 'k'; column++)
 {
 MessageBox.Show("Döngüde column değişkeninin aldığı değer " + (column));
 }

 for (int row = 1; row < 3; row++)
 {
   for (char column = 'a'; column < 'd'; column++)
   {
   MessageBox.Show("Döngüde column değişkeninin aldığı değer " + (row) + ","+(column));
   }
 }
```



Özet

- Programlama dillerinde sistematik bir akış bulunmaktadır. Sistematik akış, temel olarak tasarlanan kodun baştan sona doğru ilerlemesi ile gerçekleşir. Söz konusu akış içerisinde karar verme süreçlerinin kullanılması gereken durumlar bulunmaktadır. Karar verme mekanizması sayesinde sistematik akış farklılaşarak farklı kod bölümlerde işlem yapabilme kabiliyetine sahip olmaktadır. Ayrıca döngüler sayesinde belirlenen kod blokları tekrar tekrar çalıştırılarak, sistematik tekrarlanan akış tasarlanabilmektedir. Bazı durumlarda karar verme mekanizmaları yerine daha sade yapılar da kullanılabilmektedir.
- Karar yapıları bilgisayarın iki ya da daha fazla akış bulunması durumunda seçim yapabilmesini sağlayan mantık yapılarıdır. Karar yapıları olmadan yapılacak işlemler basit hesaplamaların ötesine geçemeyecektir. Karar mekanizması günlük yaşamımızda yaptığımız her seçime benzer özellikler sağladığından anlaşılması kolay mantıksal gruplardandır. C# programlama dilinde karar yapıları incelediği zaman karşımıza iki farklı yapı çıkmaktadır bunlar: "if" (eğer) - "else" (başka) ve "switch – case"'dir. Bu yapılar kullanılarak tek koşullu, çok koşullu ve iç içe karar yapıları oluşturulabilir.
- Programlama dillerinde tasarlanan akış birçok kez çalıştırılmak zorunda kalınabilir. Tekrarlanan akışı çalıştırabilmek için kod bloğu bir kez yazılır ve döngü ifadeleri ile tekrar tekrar çalıştırılır. Programlama dillerinin temel yapılarından olan döngüler neredeyse her dilde aynı yapıya sahiptir. C# programla dili incelendiği zaman karşımıza dört farklı döngü yapısı çıkmaktadır. Döngü yapıları "for", "while", "do-while" ve "foreach" olarak bu dil içerinde tanımlanmıştır.

DEĞERLENDİRME SORULARI

1. İki ya da daha fazla akış bulunması durumunda seçim yapabilmesini sağlayan yapıya……. mekanizması denir.

   Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?

   a) Seçim
   b) İstek
   c) Karar
   d) Bulma
   e) Anlama

2. Aşağıdakilerden hangisi karar mekanizması ifadesidir?

   a) for
   b) while
   c) as
   d) if
   e) foreach

3. Aşağıdakilerden hangisi döngü ifadesidir?

   a) Switch-case
   b) İf-else
   c) in
   d) int
   e) for

4. Switch case karar yapısında break ifadesi kullanılmazsa hangi durum gerçekleşir?

   a) Sadece ilk case ifadesi çalışır.
   b) Hiçbir case ifadesi çalışmaz
   c) Tüm case ifadeleri çalışır.
   d) Switch ifadesi çalışmaz akış devam eder.
   e) Switch ifadesi hata verir.

5. Aşağıda verilen kod bloğunda kaç döngü gerçekleşir?

```
int i = 1;
do
{
Console.WriteLine("i = {0}", i);
i++;
} while (i < 4);
```

   a) 1
   b) 2
   c) 3
   d) 4
   e) 5

6. Aşağıda verilen kod bloğunda toplam değerinin çıktısı kaçtır?

```
int toplam = 0;
int artir = 0;
do
{
  toplam += artir;
   artir++;
}
while (sayac < 4);
```

a) 4
b) 6
c) 8
d) 10
e) 12

7. Aşağıda verilen kod bloğunu çıktısı hangisidir?

```
string[] sayi= {"1", "5", "9", "4"};

foreach (string yazilacak in sayi)

{

 Console.WriteLine(yazilacak);

}
```

a) 1 5 9 4
b) 1 4 5 9
c) 9 5 1 4
d) 9 1 5 4
e) 5 1 4 9

8. Aşağıda verilen for döngüsü tanımlamalarından hangisi sonsuz döngüye girer?
   a) For (int i = 0; i<10; i++)
   b) For (int i=0; i<10; i--)
   c) For (int i= 10; i>0; i--)
   d) For (int i=0; i< 100; i=i+10)
   e) For(i=100; i>0; i=i-10)

9. Aşağıda verilen kod bloğunun çıktısı nedir?

```
for (; ;)
{
Console.WriteLine ("A");
}
```

   a) A bir defa yazar.
   b) A 10 defa yazılır.
   c) For döngüsü çalışmaz.
   d) A sonuz defa yazılır.
   e) Hiçbir şey yazmaz.

10. Tekrarlanan akışı çalıştırabilmek için kod bloğu bir kez yazılır ve …….. ifadeleri ile tekrar tekrar çalıştırılır.

    Cümledeki boşluğa aşağıdakilerden hangisi getirilmelidir?

   a) Karar
   b) Çalıştırma
   c) Operatör
   d) Koşul
   e) Döngü

Cevap Anahtarı
1.c, 2.d, 3.e, 4.c, 5.c, 6.b, 7.a, 8.b, 9.d, 10.e

YARARLANILAN KAYNAKLAR

ALGAN, Sefer, (2009), Her Yönüyle C#, Pusula Yayıncılık, İstanbul.

KIZILÖREN, Tevfik, (2013), Her Yönüyle C, KODLAB, İstanbul.

AKTAŞ, Volkan, (2013), Her Yönüyle C# 5.0, KODLAB, İstanbul.