BÖLÜM 7

DÖNGÜLER

ANAHTAR KAVRAMLAR

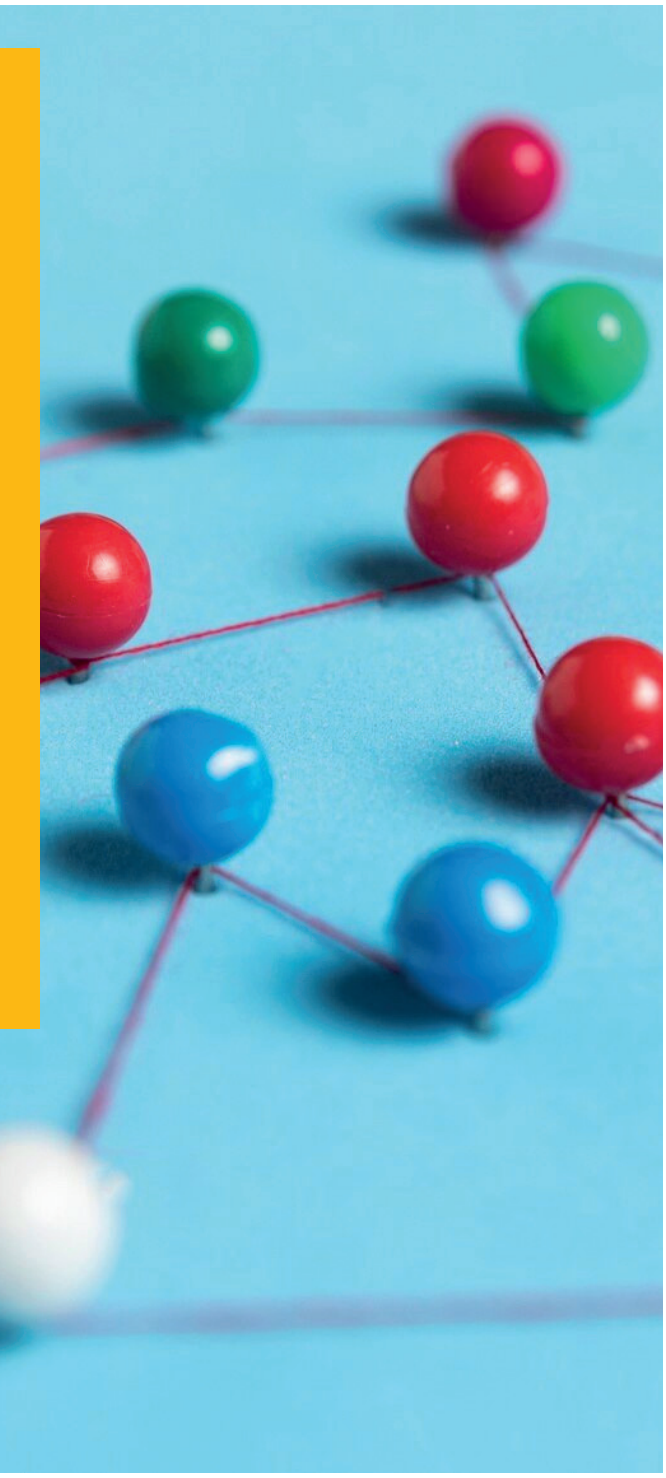
While, For, Do-While

ÖĞRENME HEDEFLERİ

1. Öğrenci bölümlere ayırmaya ilişkin temel kavramları bilir
2. Öğrenci bölümlere ayırma ilkelerini, sistemlerini ve önemini bilir
3. Öğrenci bölümlere ayırma sistemlerinin üstünlük ve sakıncalarını bilir
4. Öğrenci işletmelerde bölümlere ayırma sürecinde karşılaşılan temel sorunları algılar ve çözüm üretir
5. Aritmetik işleçlerin öncelik sıralamasını bilir

ALTINCI BÖLÜM

GİRİŞ



Döngüler, programlamının temelini oluşturan mantıksal yapılardan biridir. Döngüler, bilgisayarların belirli görevleri yinelemeli (tekrar tekrar) gerçekleştirmesini sağlar. Yineleme işlemi, programcının başlangıçta belirttiği adım sayısı kadar veya belirli bir koşul gerçekleşene kadar devam eder. Bazı durumlarda kullanıcıdan bir değer girmesi istenerek programın devamına ya da sonlandırılmasına karar verilebilir. Programlamada kullanılan çeşitli döngü yapıları vardır. Bunlar;

- for döngüsü (for-loop)
- while döngüsü (while-loop)
- do-while döngüsü (do-while loop)
- İç-içe döngüler (Nested loops) şeklinde isimlendirilir.

1. DÖNGÜLER

Bilgisayarın belirli görevleri belirli bir koşul sağlanana kadar veya başlangıçta belirtilen adım sayısı kadar yapılması döngüler vasıtasıyla sağlanır. Örneğin bir yazılım geliştiricisi bir sınıftaki her bir öğrencinin not ortalamasını bulan bir program yazmak isterse, bunu her bir öğrenci için ayrı ayrı yapmak yerine belirli kurallar dahilinde oluşturduğu formülü döngü içerisinde kullanarak bu işlemi gerçekleştirir. Böylece bilgisayar tüm öğrenciler için not ortalaması hesaplanana kadar aynı işlemi yinelemeli olarak gerçekleştirir. Döngüler programlamada zaman ve maliyetin düşürülmesi konusunda oldukça etkilidir. Bu bölümde C++ programları geliştirilirken, programları daha etkili hale getirmek için döngülerden nasıl faydalanılacağı ve hangi döngü yapısının hangi problemlerde kullanılabileceği gösterilecektir.

C++ programlama dilinde kullanılabilecek 4 temel döngü yapısı bulunmaktadır. Bunlar;

- while Döngüsü
- do-while Döngüsü
- for Döngüsü
- İç-içe Döngüler (Nested Loops)

olarak ifade edilebilirler.

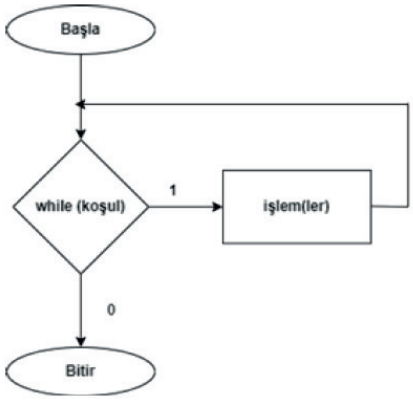
Döngüler aynı zamanda sabit döngüler ve koşullu döngüler olarak sınıflandırılabilirler. For döngüsü sabit döngü sınıfında değerlendirilirken, while ve do-while döngüleri koşullu döngülere örnek olarak verilebilir.

2. while DÖNGÜSÜ

Koşullu döngüler sınıfında değerlendirilen while döngüsünün başlatılması ve devamı belirli bir koşul doğru olduğu sürece geçerlidir. Koşul sağlanmadığında döngü başlatılmaz veya döngünün çalışması esnasında koşul bozulursa döngü sona erer.

```
while (koşul) {
   işlem(ler)
}
```

Yukarıdaki örnek kod ile while döngüsünün sözdizimi (syntax) yapısı verilmiştir. Bu yapının akış diyagramı (flowchart) ise aşağıdaki şekilde gösterilebilir.



Şekil.7 1. while döngüsü akış diyagramı.

Akış diyagramında 1 ve 0 şeklinde gösterilen ifadeler mantıksal ifadelerdir. Algoritma ve akış diyagramları oluşturulurken, evet anlamına gelen mantıksal ifadeler 1, hayır anlamına gelen mantıksal ifadeler ise 0 ile gösterilir. Bazı kaynaklarda 1 yerine true, 0 yerine ise false ifadeleri kullanılabilir. Yukarıdaki örnekte koşulun doğru (1) olması durumunda program döngü içerisinde tanımlanan işlem ya da işlemleri yapacak şekilde döngü içerisine girer. Eğer koşul başlangıçta ya da programın çalışması esnasında bozulursa program mantıksal hayır (0) sonucunu oluşturur ve program bu yönde ilerler.

Örnek.7.1. Ekrana 5 defa "Ankara Üniversitesi" ifadesini yazan C++ programını while döngüsü kullanarak yazınız.

ÖRNEK 7.1. C++ Kodlar

```
#include<iostream>
using namespace std;
int main ( ) {
  int a=0;
while (a<5)
{
cout<<"Ankara Üniversitesi\n";
a=a+1;
}
}
```

Kodların Açıklaması:

• Main fonksiyonu içerisinde tanımlanan int a=0 ifadesi döngü koşul ifadesi olarak oluşturulmuştur. Başlangıçta tanımlanan a değişkeninin değeri 0 (sıfır) olarak ayarlanmıştır.

• while (a<5) ifadesi döngünün başlatılması ve devamını kontrol edecek yapı olarak kurulmuştur. Başka bir ifade ile a değişkeninin değeri 5'den küçük olduğu sürece döngü başlatılacak ve bu koşul sağlandığı sürece döngü içerisindeki işlem(ler) gerçekleştirilecektir. Örnekte a değişkeninin başlangıç değeri 0'dır ve 0 (sıfır) 5'den küçük olduğu için program döngü içerisine girecektir.

• cout<<"Ankara Üniversitesi\n" döngü denetiminden sonra ilk çalıştırılcak kod satırıdır ve ekrana "Ankara Üniversitesi" ifadesini yazdıracaktır.

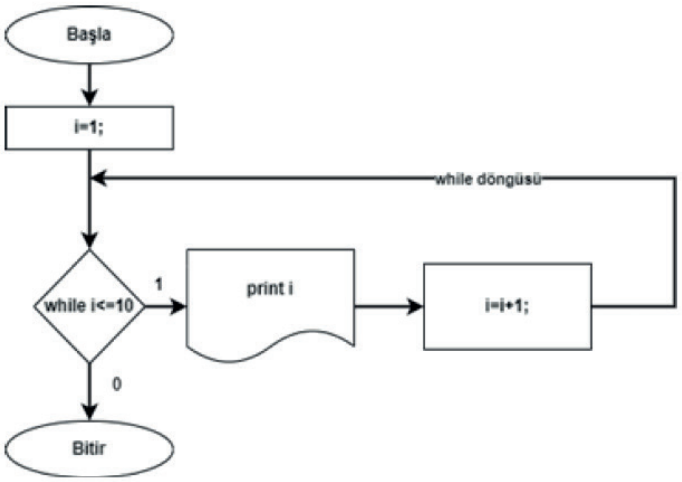
• a=a+1 ifadesi döngü denetimini yapmak için kullanılan ifadedir. Programlamada sayaç (counter) olarak bilinir. Program bu satıra geldiği anda a değişkeninin değeri 1 artırılacaktır. Böylece a değişkeni değerinin sürekli 5 değerinden küçük olmasının önüne geçilecektir. Bu denetimin yapılmadığı durumlarda program sonsuz (infinite loop) döngüye girecektir ve asla döngü dışına çıkamayacaktır. Döngü değişkeni a'nın değeri 1 artırıldıktan sonra program tarafından a'nın değerinin tekrar 5 değerinden küçük olup olmadığı kontrol edilecek, eğer küçük ise program yeniden döngü içerisindeki işlem(ler)i gerçekleştirecektir. Döngü değişkeni a'nın değeri 5' e ulaştığında bu denetim sonucu hayır (0) mantıksal cevabını oluşturacak ve program döngü yapısı içerisinden çıkacaktır.

Yukarıda açıklaması verilen kodların çalışma mantığı tablo görünümünde aşağıdaki şekilde ifade edilebilir.

Yineleme (Iteration)	Değişken Değeri	Koşul (a<5)	İşlem(ler)
1.	a=0	Evet (1)	Ankara Üniversitesi
2.	a=1	Evet (1)	Ankara Üniversitesi
3.	a=2	Evet (1)	Ankara Üniversitesi
4.	a=3	Evet (1)	Ankara Üniversitesi
5.	a=4	Evet (1)	Ankara Üniversitesi
6.	a=5	Hayır (0)	Döngü Dışı

Örnek.7.2. 1'den 10'a kadar olan sayıları ekrana yan yana yazdıran C++ kodlarını while döngü yapısı ile yazdırınız.

İstenilen programın akış diyagramı Şekil.7.2'de verilmiştir. Bu örnekte döngü değişkeni olarak i seçilmiştir. Örnekte 1'den 10'a kadar olan sayıların ekrana yan yana yazdırılması istenildiği için i değişkeninin başlangıç değeri 1 olarak belirlenmiştir. Program döngü denetimi noktasına geldiğinde öncelikle i değişkeninin değerinin küçük-eşit (<=) 10 olup olmadığı kontrol edilecek, eğer değer bu koşula uyuyorsa program döngü içerisine girecektir. Döngü içerisinde öncelikle i değişkeninin değeri ekrana yazdırılacak arkasından değişkenin değeri 1 artırılacaktır. Program bu noktadan sonra i değişkenin yeni değeri ile yeniden döngü denetimin yapıldığı satıra dönecek ve değişken değerinin 10'a eşit ya da küçük olma durumu yeniden kontrol edilecektir. Değer küçük-eşit 10'olduğu sürece program tekrar-tekrar döngü içerisine girecek ve buradaki işlemleri gerçekleştirmeye devam edecektir.



Şekil.7 2 while döngüsü (devam).

ÖRNEK 7.2. C++ Kodlar

```
#include<iostream>
using namespace std;
int main ( ) {
   int i=1;
while (i<=10)
{
cout<<i<<" ";
i=i+1;
}
}
```

Çalışma Sorusu: Örnek 7.2 için 1'den 10'a kadar olan sayıları alt-alta yazdırmak için kodda nasıl bir değişiklik yapılmalıdır?

break Komutu: Bazı durumlarda döngü koşulu bozulmadan istenilen bir sonuca ulaşıldığında döngünün sonlandırılması gerekebilir. C++'da break komutu döngüyü sonlandırmak için kullanılan bir kontrol ifadesidir. Döngü içerisinde herhangi bir noktada program break komutu ile karşılaşırsa döngüyü o noktada sonlandırır ve kontrolü döngüden sonraki ilk ifadeye bırakır. Bu komut while, do-while, for, iç-içe döngüler ve sonsuz döngülerde kullanılabilir. Bu durum Örnek 7.3 üzerinden incelenebilir.

Örnek 7.3. Klavyeden girilen sayıların karekökünü hesaplayan C++ programını while döngüsü ile oluşturunuz. Eğer klavyeden negatif bir sayı girilirse program sonlandırılmalıdır. Bu örnek için sonsuz döngü (infinite loop) yapısı oluşturulabilir. Program kullanıcının klavyeden girdiği sayının karekökünü hesaplayacak ve sonucu ekrana yazacaktır. Bu durum kullanıcı negatif bir sayı girene kadar devam edecektir. Örnekte karekök hesaplaması yaptırılacağı için sqrt () fonksiyonun içerisinde bulunduğu <cmath> header dosyasının programa eklenmesi gerekmektedir.

ÖRNEK 7.3. C++ Kodlar

```
#include<iostream>
#include<cmath>
using namespace std;
int main ( ) {
double sayi, sonuc;
while (true) {
      cout<<"Karekökü hesaplanacak sayıyı giriniz:";
      cin>>sayi;
      if (sayi<0)
         break;
      cout<<sayi<<" sayısının karekökü:"<<sqrt(sayi)<<endl;
      }
cout<<"Negatif bir sayı girildiği için döngü sonlandırılmıştır!!!";
}
```

• Program sonsuz döngü yapısı ile oluşturulurken döngü devam koşulu için "true" mantıksal değeri kullanılmıştır. Böylelikle while döngüsü başlangıç koşul sorusunun cevabı varsayılan olarak evet (true) şeklinde kabul edilmiştir. Bu örnekte döngü denetimi break komutu kullanımı ile gerçekleştirilmiştir.

• Program içerisinde kullanılan sayi ve sonuc isimli değişkenler double veri türünde oluşturulmuştur. Böylelikle ondalıklı sayı türlerinde işlem yapılabilmesi sağlanmıştır.

continue Komutu: Döngü yinelemesi (iteration) esnasında program continue komutu ile karşılaşırsa döngünün bu komuttan sonra gelen ifadeleri değerlendirilmez ve döngü bir sonraki yineleme adımı ile yeniden başlatılır. Komut while, do-while, for ve iç-içe döngü yapılarında kullanılabilir. Bu durum aşağıdaki örnek üzerinden incelenebilir.

Örnek 7.4. Bir ders için öğrenci ortalamasını hesaplayan C++ programını while döngüsü yapısı ile oluşturunuz. Eğer girilen not vize ya da final sınavı için 100'den büyük ya da 0'dan küçükse kullanıcıdan yeni değer girmesi istenilmelidir. Programdan çıkış için C değerinin girilmesi gerekmektedir.

```
ÖRNEK 7.4. C++ Kodlar
#include<iostream>
using namespace std;
int main ( ) {
int vize, final;
double ortalama;
char cikis;
while (true) {
      cout<<"Öğrencinin Vize Notunu Giriniz:";
      cin>>vize;
      if (vize>100 || vize<0){
        cout<<"Geçerli Bir Değer Giriniz (0-100 arası):";
        continue;
      }
      cout<<"Öğrencinin Final Notunu Giriniz:";
      cin>>final;
      if(final>100 || final <0){
        cout<<"Geçerli Bir Değer Giriniz (0-100 arası):";
        continue;
      }
      ortalama=vize*0.3+final*0.8;
      cout<<"Öğrencinin Ortalaması="<<ortalama<<endl;
      cout<<"Programdan Çıkış İçin C, Devam İçin D Giriniz:";
      cin>>cikis;
      if (cikis=='C')
        break;
 }
cout<<"Program kullanıcını isteği ile sonlandırılmıştır";
}
```

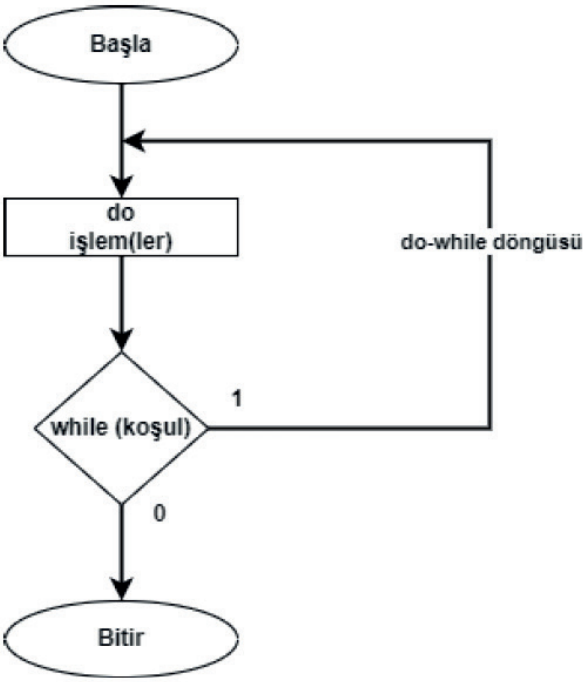
Önemli Not: C++ büyük-küçük harf duyarlı (Case Sensitive) bir programlama dilidir. Dolayısıyla programdan çıkış yapabilmek için büyük harf C'nin girilmesi gerekmektedir. Örnek için öğrencinin ortalaması vize notunun %30'u final sınavının %80'i alınarak hesaplattırılacaktır. Program continue komutu ile karşılaşınca döngü başından itibaren yeniden çalıştırılırken break komutu ile karşılaştığında döngü sonlandırılmaktadır. Bu örnek içinde sonsuz döngü yapısı kullanılmıştır.

3. do-while DÖNGÜSÜ

do-while döngüsü while döngü yapısına oldukça benzemektedir. Bu yapıda döngü denetimi döngü başında değil döngü sonunda gerçekleştirilir. Bu özelliği yüzünden döngü içerisindeki ifadeler en az bir kez çalıştırılır. Bu döngünün syntax yapısı aşağıdaki şekilde verilir.

```
do {
   işlem(ler)
} while (koşul);
```

Bu yapı için akış diyagramı ise aşağıdaki şekilde oluşturulabilir.



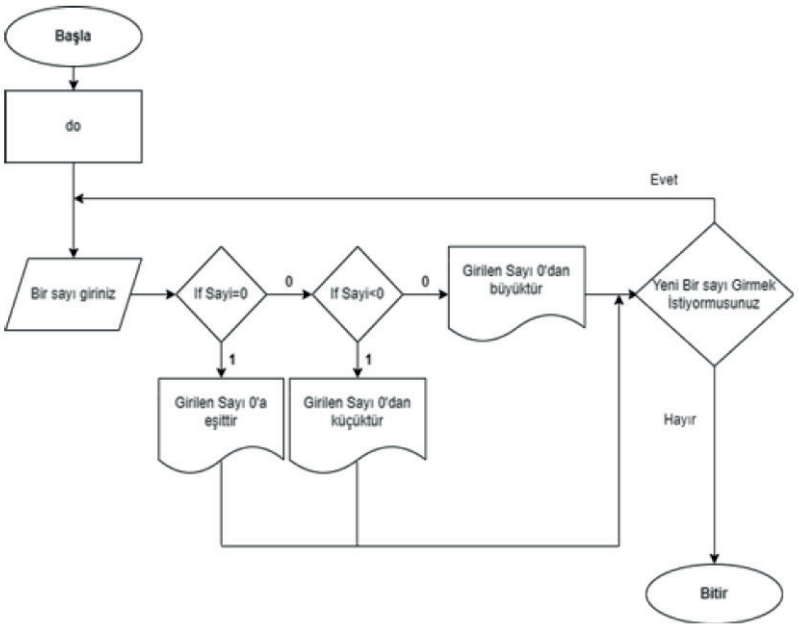
Şekil.7 3. do-while döngüsü akış diyagramı.

Akış diyagramından da rahatlıkla görülebileceği üzere döngü içerisindeki işlem(ler) en az bir kez çalıştırılır ve döngü denetimi bu işlem sonrasında gerçekleştirilir.

Örnek 7.5 üzerinden do-while ile while döngü yapısının arasındaki farklılıkları inceleyiniz.

Örnek.7.5. Klavyeden girilen bir sayının 0'dan küçük, 0'dan büyük ya da 0'a eşit olup olmadığını bulan C++ kodlarını do-while döngü yapısı yazınız.

Not: Program kullanıcıdan gelecek cevaba göre (Evet (E), Hayır (H)) çalışmaya devam edecek ya da sonlandırılacaktır.



Şekil.7 4. Örnek7.5 için akış diyagramı.

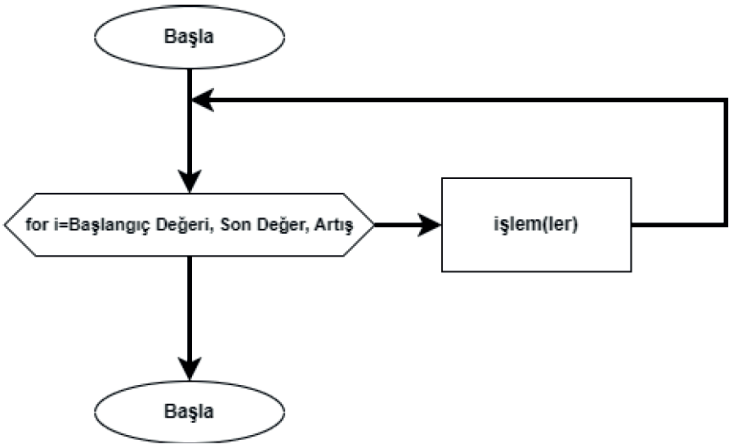
```
ÖRNEK 7.5. C++ Kodlar
#include<iostream>
#include<string>
using namespace std;
int main ( ) {
  int sayi;
  char devam;
  do {
      cout<<"Bir Sayı Giriniz:";
      cin>>sayi;
      if (sayi==0)
        cout<<"Girilen Sayı 0'a eşittir\n";
      if (sayi<0)
        cout<<"Girilen Sayı 0'dan küçüktür\n";
      if (sayi>0)
        cout<<"Girilen Sayı 0'dan büyüktür\n";
cout<<"Yeni Bir Sayı Girmek İstiyor musunuz (Evet için E, Hayır için H'ye Basınız):";
 cin>>devam;
      } while(devam=='E');
}
```

4. for Döngüsü

Programlamada kullanılan bir başka döngü yapısı for döngüsüdür. Sabit döngü türlerine örnek olarak gösterilebilir. Şartlı (koşullu) döngülerden farkı döngünün başlangıcı ve devamı için belirli bir koşulun gerçekleşmesine ihtiyaç duymamasıdır. Standart for döngüsü kullanımında döngü başlangıç değeri (initial value), son değer (final value) ve artış miktarı (increment) tanımlamaları ile döngünün yinelemeli işlemleri kaç defa yapacağı döngü başında belirtilir. Döngünün syntax yapısı aşa-ğıdaki şekilde verilebilir.

```
for (Başlangıç Değeri; Son Değer; Artış) {
   işlem(ler)
}
```

Döngünün başlangıç değeri artış miktarı ve son değeri arasında ";" karakterinin kullanılması gerekmektedir. Bu döngü yapısında döngü değişkeni ve değişken türü döngü başlangıç kısmında tanımlanır. Akış diyagramlarında while ve do-while döngüsünden farklı olarak aşağıdaki gibi bir sembol ile ifade edilirler.



Şekil.7 5. For döngüsü için akış diyagramı

Örnek 7.6. Kullanıcı tarafından klavyeden girilen sayı değerine kadar bütün sayıları toplayıp (1'den n'e kadar) sonucu ekrana yazan C++ programını for döngü yapısını kullanarak yazınız.

```
ÖRNEK 7.6. C++ Kodlar
#include<iostream>
using namespace std;
int main ( ) {
int sayi, toplam;
toplam=0;
cout<<"1'den Büyük Bir Sayı Giriniz:";
cin>>sayi;
  for (int i=1; i<=sayi; i++){
     toplam +=i;
     }
cout<<"1 ile "<<sayi<<" arasındaki sayıların toplamı="<<toplam<<endl;
}
```

Örnekte öncelikle kullanıcının gireceği sayı ve sonuc için tamsayı (integer) türünde değişken tanımlamaları yapılmıştır. Örnekte döngü değişkeni i'nin tanımlanması (int i=1) döngü içerisinde gerçekleştirilmiştir. Bu şekilde bir uygulama geliştirildiğinde döngü değişkeninin etki alanın sadece for döngüsü içerisinde olduğu unutulmamalıdır. Başka bir ifade ile döngü dışında i değişkeni değerine erişilemez. Örneğin döngü değişkeni olarak kullanılacak değişken main fonksiyonu etki alanında tanımlanmış olsaydı main fonksiyonun herhangi bir noktasından erişilebilir olurdu. Ancak bu kullanımın başka problemlere yol açabileceği unutulmamalıdır1.

Örnek 7.7. Klavyeden girilen bir sayının asal olup olmadığını bulan C++ programını for döngüsü kullanarak yazınız.

```
ÖRNEK 7.7. C++ Kodlar
#include<iostream>
using namespace std;
int main ( ) {
   int sayi;
   int sayac=0;
   cout<<"Bir Sayı Giriniz:";
   cin>>sayi;
   for (int i=1;i<=sayi;i++) {
        if(sayi % i ==0)
          sayac=sayac+1;
   }
  if (sayac==2)
    cout<<"Girilen Sayı Asaldır";
  else
    cout<<"Girilen Sayı Asal Değildir";
}
```

Ders video kaydında bu durum detaylı bir şekilde ele alınacaktır

Program yazılırken asal sayı tanımından yola çıkılmıştır. Sadece 1'e ve kendisine tam olarak bölünebilen sayılar asal sayılardır. Başka bir ifade ile asal sayıların 1 ve kendisi olmak üzere 2 adet tam böleni vardır. Döngü içerisinde tanımlanan if (sa-yi%i==0) ifadesi ile klavyeden girilen sayının 1'den başlayıp kendisine kadar bütün sayılarla tam olarak bölünüp bölünmediğinin kontrolü yapılmaktadır. Burada kullanılan % aritmetiksel operatörü aynı zamanda kalan (modulo) operatörü olarak bilinir. Bir sayının diğer sayıya bölümünden kalanını verir. Örnekte bu kalanın 0'a eşit olup olmadığı kontrol edilmektedir.

Her tam olarak bölünme işlemi sonrasında sayaç değişkeni değeri bir artırılmaktadır. Döngü dışında yeni bir kontrol mekanizması oluşturularak eğer sayaç değişkeninin değeri 2 ise (sadece 2 sayı ile tam olarak bölünmüştür) bu sayı için "Girilen Sayı Asaldır" ifadesi ekrana yazdırılmıştır. Eğer sayaç değişkeninin değeri 2'den farklı ise klavyeden girilen sayıyı tam olarak bölen 2'den fazla sayı olacaktır ve bu durum asal sayı tanımına uymayacaktır.

For döngüsünün farklı bir kullanımı için Örnek 7.8'i inceleyiniz.

ÖRNEK 7.8. C++ Kodlar

```
#include<iostream>
using namespace std;
int main ( ) {
   for (;;) {
      cout<<"Ankara Üniversitesi\n";
   }

}
```

Bu örnekte, döngü başlangıç değeri, son değer ve artış miktarı belirtilmemiştir. Döngü bu haliyle sonsuz bir döngü formuna dönüşmüştür. Başlangıçta program döngü içerisine girer ve manuel olarak sonlandırılmadığı sürece (Ctrl+C) "Ankara Üniversitesi" ifadesini alt-alta yazar.

5. İç-içe Döngüler (Nested Loops)

Karar ifadelerinde (if) olduğu gibi döngü yapılarının da iç-içe oluşturulması mümkündür. Başka bir ifade ile bir döngü içerisinde başka döngü yapıları oluşturulabilir. Bu yapıyı daha iyi anlamak için aşağıdaki örneği inceleyiniz.

Örnek 7.9. Aşağıdaki ekran görüntüsünü verecek olan C++ programını iç-içe for (nested for) döngü yapısını kullanarak yazınız.

Ekran görüntüsü 3 satır, 3 sütun * (yıldız) karakterlerinden oluşturulmuştur. Bu ekran görüntüsünü elde edebilmek için iç-içe döngü yapısının kullanılması gerekmektedir.

ÖRNEK 7.9. C++ Kodlar

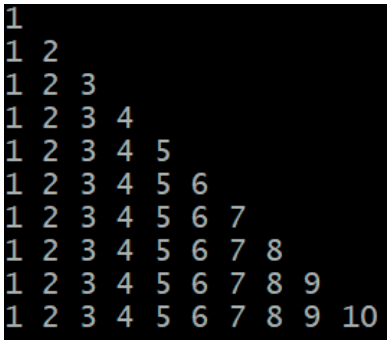
```
#include<iostream>
using namespace std;
int main ( ) {
   for (int i=1;i<=3;i++) {
     for (int j=1;j<=3;j++){
      cout<<"*";
     }
    cout<<"\n";

   }
}
```

Örnek 7.9'dan görülebileceği gibi içteki döngü dıştaki döngünün etki alanı içerisindedir. Program ilk çalıştırıldığında önce dıştaki döngü ve hemen ardından içteki döngü başlatılır. Bu aşamada i değişkeni 1 değerine sahip iken j değişkeni 1'den 3'e kadar değerler alır. J değişkenin her bir değerinde ekrana bir adet "*" karakteri basılır. Bu karakterler cout ifadesinde "/n" veya "endl" kullanılmadığı için yan yana yazdırılır. İçteki döngünün son değerinden sonra kontrol dıştaki döngüye geçer ve bu döngünün etki alanındaki cout<<"/n" ifadesi çalıştırılır ve ekranda bir alt satıra geçilir. Bu işlemin ardından i değişkeninin ikinci değeri (2) için döngü yeniden başlatılır. İçteki döngü yeniden başlatılırken j değişkeni yeniden 1 değerini alır ve bu işlem toplam 3 satır ve 3 sütun "*" karakterlerinin ekrana basılma işleminin sonuna kadar devam eder. Aşağıdaki tabloda bu durum ifade edilmiştir.

i=1	j=1	*
	j=2	**
	j=3	***
i=2	j=1	*** *
	j=2	*** **
	j=3	*** ***
i=3	j=1	*** *** *
	j=2	*** *** **
	j=3	*** *** ***

İç-içe döngü yapısında farklı döngüler birlikte kullanılabilir. Örneğin while döngüsü içerisinde for döngüsü, onun içerisinde do-while döngüsü gibi. Bu durum aşağıdaki örnek üzerinden incelenebilir.

Örnek 7.10. Aşağıdaki ekran görüntüsünü veren C++ kodlarını for ve while döngülerini iç-içe (nested) yapıda kullanarak kodlayınız.

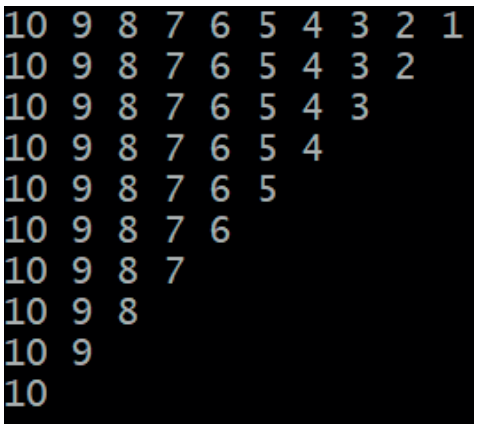


ÖRNEK 7.10. C++ Kodlar

```
#include<iostream>
using namespace std;
int main ( ) {
      for (int i=1;i<=10;i++)
      {
       int sayi=1;
              while(sayi<=i) {
                cout<<sayi<<" ";
                sayi+=1;
              }
       cout<<"\n";
      }
}
```

Bir programı yazmanın birden fazla yolu olabileceği unutulmamalıdır. Programlamada öğrenilen her yeni özellik ile program daha farklı şekillerde yazılabilir. Bu örnekte for döngüsü içerisinde (etki alanında) while döngüsü kullanılmıştır. Program sadece for döngüleri ya da while döngüleri ile oluşturulabilirdi. Örnekte, for döngüsü etki alanında sayı adında bir değişken kullanılarak başlangıç değeri 1 olarak atanmıştır (Bu değer for döngüsünün her yinelemesinde yeniden 1 olacak şekilde düzenlenmiştir). İçteki döngü while yapısı ile oluşturulmuştur. Bu yapı ile sütun sayısı, dıştaki döngünün o anki değeri kullanılarak dinamik şekilde belirlenmiştir. Buradaki koşul (sayi<=i) ile sayının değeri i'den küçük olduğu sürece sayının alacağı değerler ekrana aralarında bir boşluk eklenerek yazdırılmıştır. Döngü denetimi yine sayı isimli değişken değeri ile sağlanmaktadır.

Çalışma Sorusu: Örnek 7.10'u aşağıdaki ekran görüntüsünü verecek şekilde düzenleyiniz.



BÖLÜM ÖZETİ

Döngüler, programlama dillerinde yinelemeli işlemlerin gerçekleştirilmesinde temel bir role sahiptir. Programlarda tekrar eden işlemlerin etkin bir şekilde uygulanması gerektiğinde döngü yapıları kullanılmaktadır. Döngüler, programcılara daha esnek ve dinamik bir kod geliştirme imkânı sunarak, karmaşık algoritmaların daha sade bir biçimde ifade edilmesine olanak tanır.

Programlama süreçlerinde döngüler, sabit veya koşullu yapılar şeklinde tercih edilebilir. **Koşullu döngüler**, belirli bir koşul sağlandığı sürece çalışmaya devam eden yapılar olarak tanımlanır. Bu koşul, döngü oluşturulurken sabit bir değer olarak belirlenebileceği gibi, programın çalışma esnasında kullanıcı tarafından girilen dinamik bir değerle de şekillendirilebilir. **Sabit döngüler** ise döngünün yineleme sayısının önceden tanımlandığı yapılardır. Bu tür döngüler, belirli sayıda işlem gerçekleştirilmesi gerektiği durumlarda kullanılmaktadır.

Döngü yapıları, algoritmaların verimli bir şekilde uygulanması ve programların performansının artırılması açısından programlama dillerinin vazgeçilmez bir bileşenidir.

GÖZDEN GEÇİRELİM

1. **Belirli bir koşul altında, bir döngüyü sonlandırmak için aşağıdaki komutlardan hangisi kullanılır?**
   a) continue
   b) quit
   c) leave
   d) go
   e) break

2. **Adım sayısı bilinen yinelemeli bir işlem için aşağıdaki döngülerden hangisi kullanılmalıdır?**
   A) for
   b) while
   c) do-while
   d) while-while
   e) do-while-while

3. **Aşağıdaki döngülerden hangisi döngünün devam koşulunu döngü sonunda kontrol eder?**
   A) for
   b) while
   c) do-while
   d) for-while
   e) for-for

4. **Program döngü içerisinde continue komutu ile karşılaşırsa aşağıdaki ifadelerden hangisi(leri) gerçekleşir.**
   I. Komuttan sonra gelen hiçbir ifade çalıştırılmaz döngü bir sonraki değer ile çalışmaya devam eder
   II. Komuttan sonra gelen ifadeler bir defa çalıştırılır ve döngü sonlandırılır
   III. Döngü hemen sonlandırılır
   a) Yalnız I
   b) Yalnız II
   c) Yalnız III
   d) II ve III
   e) Hiçbiri

5. **Aşağıdaki program çalıştırıldığına aşağıdakilerden hangisi gerçekleşir?**

```
#include<iostream>
using namespace std;
int main ( ) {
    for(int i=1;i<10;i++){
      if(i==5 );
        break;
   }
cout<<i;
}
```

   a) 1 2 3 4 5
   b) 1 2 3 4 5 6 7 8 9 10
   c) 5 10
   d) 5
   e) Program derleme zamanında i değişkeni döngü etki alanı dışında olduğu için hata verir.

6. Aşağıdaki program çalıştırıldığında ekranda gözükecek ifade aşağıdakilerden hangisidir?

```
#include<iostream>
using namespace std;
int main ( ) {
    for(int i=1;i<=10;i++){
      if(i==5 || i==7)
          continue;
      cout<<i<<" ";
    }
}
```

a) 1 2 3 4 5 6 7 8 9 10
b) 1 2 3 4 6 8 9 10
c) 1 2 3 4 5
d) 5 7
e) 5

7. Aşağıdaki program çalıştırıldığında ekranda gözükecek ifade aşağıdakilerden hangisidir?

```
#in/895clude<iostream>
using namespace std;
int main ( ) {
int a=0;
int b;
      while(a<3){
      b=1;
        while(b<3){
              cout<<a<<" "<<b<<endl;
              b+=2;
        }
       a=3;
     }
}
```

a) 0 1 3
b) 0 0 0
c) 0 2 3 3
d) 0 1
e) 0 1 1 2 2 2 3 3 3 3

8. Aşağıdaki program çalıştırıldığında ekran gözükecek ifade aşağıdakilerden hangisidir?

```
#include<iostream>
using namespace std;
int main ( ) {
      for(int i=5;i>3;i--){
      cout<<i<<" ";
   }
}
```

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

9. Aşağıdaki programda verilen while döngüsü kaç defa çalışır?

```
#include<iostream>
using namespace std;
int main ( ) {
int i=0;
      while(i){
      cout<<"Ankara\n";
      i++;
      }
}
```

a) Sonuz döngü (infinite-loop)
b) Koşul sağlanmaz program döngü içine girmez
c) 1
d) 2
e) 3

10. Aşağıdaki ifadelerden hangisi bir döngü yapısı değildir?

I. for
II. while
III. if
IV. do-while

a) Yalnız I
b) Yalnız II
c) Yalnız III
d) Yalnız IV
e) Hiçbiri

Yanıt Anahtarı: 1-A, 2-A, 3-C, 4-A, 5-E, 6-B, 7-D, 8-D, 9-B, 10-C