# DÖNGÜLER

GİRİŞ

`for` Döngüsü

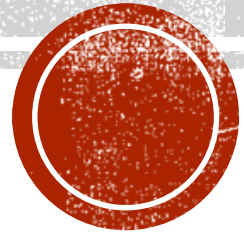
Çalışma Biçimi

`while` Döngüsü

Önemli Not

`break` Komutu

Örnekler





# GİRİŞ

- Bir çok uygulamada belirli işlemlerin tekrar tekrar gerçekleştirilmesi gerekir.
- Programlamada bu işlemler grubunu çok sayıda tekrar etmek imkanı sağlayan yapılara ÇEVRİM, DÖNGÜ veya LOOP denir.
- Çevrim, tekrarlı bir işlem yapısıdır.

Temelde iki farklı döngü yapısı vardır:

1. Çevrimdeki işlem sayısını önceden belirleyerek ve bu sayıya ulaşıp ulaşmadığını bir sayaç ile denetleyerek gerçekleştirilen çevrim yapıları

2. Çevrimin sona ermesini bir koşula bağlı olarak kontrol eden çevrim yapıları

# for DÖNGÜSÜ

- Bu döngü yapısında önceden döngünün kaç defa döneceği bilinmelidir.
- Döngü bir sayaç kullanılarak kontrol edilir.
- Sayacın kaçta başlayacağı, kaçta biteceği ve her döngüde ne kadar artacağı verilmelidir.
- Sayacın artış değeri verilmez ise 1 alınarak işlem yapılır.
- Genel yapısı aşağıdaki gibidir:

```
for  sayac = baslangıc : artıs: bitis

     işlemler

end
```




# ÇALIŞMA BİÇİMİ:

**Döngüde sayacın başlangıç değerine her seferinde artış eklenerek bitiş değerine gelene kadar döngü çevrilir. Aşağıdaki örneklerde bu durum gösterilmektedir.**

```
for  i = 1 : 2: 8
  fprintf('i= %d\n',i)
end
```

```
----------------------
i= 1
i= 3
i= 5
i= 7
```

```
for  i = 1 : 8
  fprintf('i= %d\n',i)
end
```

```
----------------------
i= 1
i= 2
i= 3
i= 4
i= 5
i= 6
i= 7
i= 8
```

```
for  i = 8 :-1: 1
  fprintf('i= %d\n',i)
end
```

```
----------------------
i= 8
i= 7
i= 6
i= 5
i= 4
i= 3
i= 2
i= 1
```

# ÖNEMLİ NOT :

**Eğer sayacın değeri dikkatli verilmez ise bazı durumlarda döngü hiç dönmez.**

**Örneğin:**

```
for  i = 1 : -1: 8
  fprintf('i= %d\n',i)
end
```

---

Ekrana hiçbir şey yazmaz. Çünkü Artış negatif olmasına rağmen, başlangıç değeri bitiş değerinden küçüktür. Döngü hiç dönderilmez

```
for  i = 8 : 1
  fprintf('i= %d\n',i)
end
```

---

Ekrana hiçbir şey yazmaz. Çünkü Artış verilmediği için 1 alınır. Ancak başlangıç değeri bitiş değerinden büyük olduğu için Döngü hiç dönderilmez




# ÖRNEK:

**Klavyeden girilen bir sayının kendisine kadar olan sayıların toplamını veren bir program yazınız:**

```
clc
clear all
n=input('bir sayi giriniz:\n');
toplam=0;
for  i = 1 : n
  toplam=toplam+i;
end
fprintf('toplam= %d \n',toplam)
```

```
Bu örnekte toplam değişkeni
döngüden önce sıfır alınır.
Ancak döngü başladıktan
sonra eski değerinin
üzerine her defasında i'nin
güncel değeri eklenir.
Girilen 3 değeri için Sonuç:
--------------------------
3
toplam= 6
```

# while DÖNGÜSÜ

- Önceden belirlenmiş belli bir ***durum*** gerçekleştiği sürece döngü devam ettirilir. Durum sağlamadığı anda döngü bitirilir.

```
while   durum

        işlemler

end
```

```
i=1;
while   i<=8
  fprintf('i= %d\n',i)
  i=i+2;
End
```

```
-----------------------
i= 1
i= 3
i= 5
i= 7
```

# ÖNEMLİ NOT :

Eğer ***durum*** her zaman doğru olacak şekilde verilirse, sonsuz döngü oluşturulur.

Örneğin:

```
while   5<8

        işlemler

end
```

Böyle bir program çalıştırılmışsa, bu program ancak Ctrl tuşuyla beraber C tuşuna basılarak bitirilebilir.

Ya da Pause ve Quit düğmelerine basılır.

# break İFADESİ

- **for** ve **while** döngülerine ilaveten bir program akışını kontrol edebilmenin diğer bir yolu **break** ifadesini kullanmaktır.
- **break** ifadesini döngü gövdesi içerisinde kullanmak, döngünün durmasını ve döngüden sonra gelen ilk ifade veya komutun işletilmesini sağlar.

```
for j=2:6
    fprintf('j= %g \n', j)
    if  j==4
        break
    end
end
disp('dongu sonlandirildi');
```

```
j=2
while j<=6
    fprintf('j= %g \n', j)
    if  j==4
        break
    end
    j=j+1;
end
disp('dongu sonlandirildi');
```