



BİLGİSAYAR PROGRAMLAMA 1

Ders Notu 6 – Döngüler

Konya Teknik Üniversitesi
Elektrik – Elektronik Mühendisliği Bölümü

21.03.2024
Konya

Döngü Deyimleri

Döngü deyimleri, belirli bir kod parçasının tanımlanmış bir koşul altında tekrarlı bir şekilde çalıştırılmasını sağlar. Kod uzunluğunu azaltarak etkinliği de artırır.

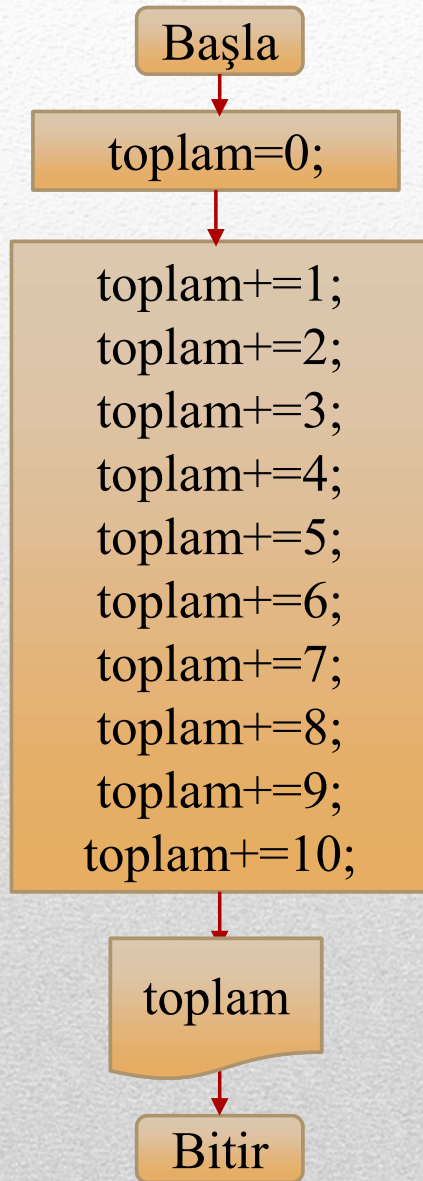
Döngünün sonlanabilmesi için, bir aşamasında mutlaka şart ifadesi geçersiz olmalıdır. Aksi takdirde sonsuz döngü oluşur.

Koşulu başta sınayanlar: **for döngüsü, while döngüsü**

Koşulu sonda sınayanlar: **do while döngüsü**

* **for** ve **while**' da çevrim içine hiç girilmeyebilir, ancak **do while**' da çevrim en az bir kez yürütülür.

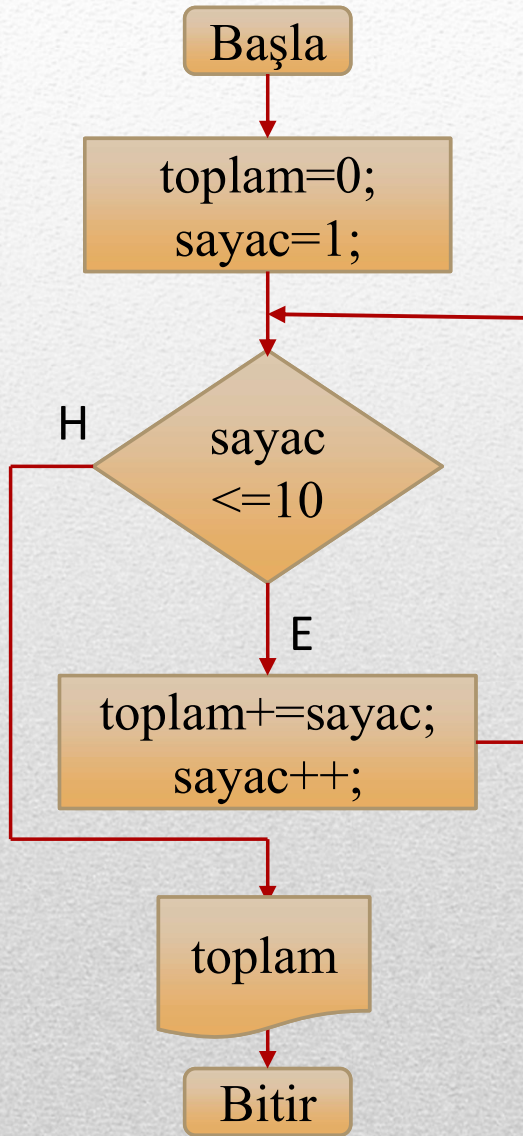
Örnek: 1'den 10'a kadar sayıların toplamını döngü kullanmadan bulduran bir kod yazınız.



```
#include<stdio.h>
int main() {
int toplam=0;
toplama=toplama+1; /* toplama+=1; */
toplama=toplama+2; /* toplama+=2; */
toplama=toplama+3;
toplama=toplama+4;
toplama=toplama+5;
toplama=toplama+6;
toplama=toplama+7;
toplama=toplama+8;
toplama=toplama+9;
toplama=toplama+10;
printf(“Sayıların toplamı: %d\\n”, toplam);

return 0; }
```


Örnek: 1'den 10'a kadar sayıların toplamını döngü kullanarak bulduran bir kod yazınız.



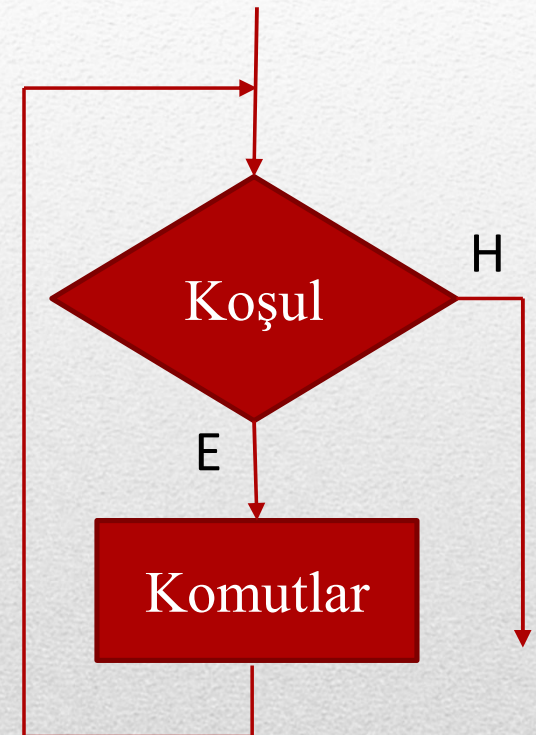
```
#include<stdio.h>
int main()
{ int toplam=0; int sayac=1;
while (sayac<=10)
{
    toplam=toplam+sayac;
    sayac++;
}
printf("Sayıların toplamı: %d\n", toplam);
return 0;
}
```

Sayıların toplamı: 55

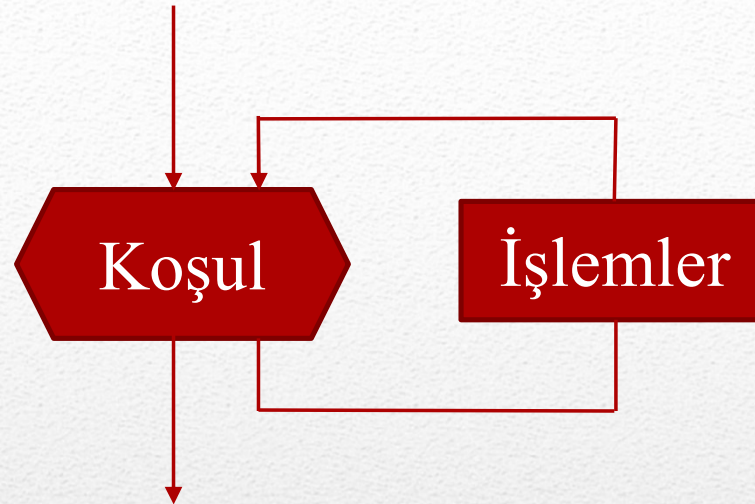
while Döngüsü

- İçine yazılan komut veya komut kümesini, belirli bir şart sağlandığı sürece tekraren çalıştırır.

```
while(koşul)
{
    işlem-1;
    ....
    işlem-N;
}
```



- İki ya da daha fazla koşulun mantıksal operatörlerle birleştirilerek verilmesi de mümkündür.

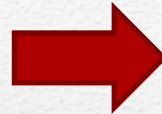


- Döngü şartı başlangıçta test edilir ve şartın sağlanması halinde döngüye girilir. Döngü içinde tek komut çalıştırılacaksa süslü parantezler kullanılmayabilir. Ancak kaç komut olursa olsun parantez kullanmayı alışkanlık haline getirmek daha doğru bir yaklaşım olacaktır. Bu yönüyle **if** yapısına benzer.

while(koşul)
Tek-Komut;

- Kullanılan sayaç değişkenine mutlaka başlangıçta bir ilk değer ataması yapılmalıdır.

```
main() {  
  int sayac;  
  while(sayac<100) {  
    printf("%d",sayac);  
    sayaç++; } }
```



Ekrana sayı değeri yazdırılıp yazdırılmayacağı ya da hangi sayıdan başlayarak 100'e kadar sayıların yazdırılacağı belirsizdir.

- Döngü içerisinde koşul değişkeninin artırılması gerektiği unutulmamalıdır.

```
main() {  
  int k=1;  
  while(k<2) {  
    printf("%d",k);  
  } }
```



Koşul değişkeni k döngü içerisinde artırılmadığı için ekrana sonsuza kadar 1 yazar.

- **Örnek:** Ekranda aşağıdaki çıktıyı oluşturmak için **while** döngüsü kullanarak bir C kodu yazmaya çalışın.
(ipucu:2 adet sayaç kullanın)

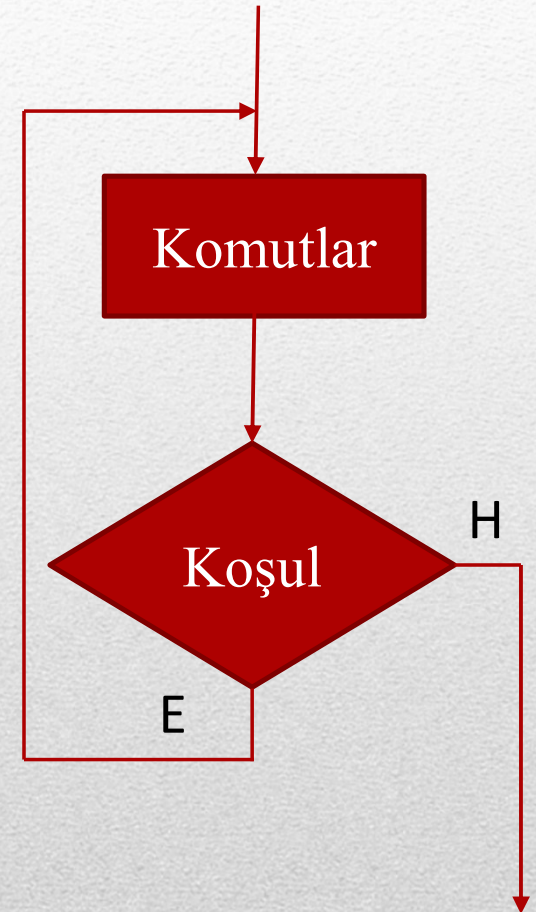
```
*  
**  
***  
****  
*****  
-----
```

```
*  
**  
***  
****  
*****
```

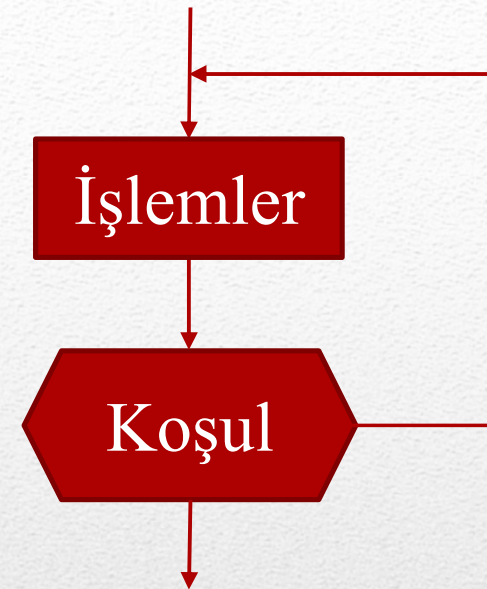
```
#include<stdio.h>  
int main()  
{  
    int i=0, j;  
    while(i<5)  
    {  
        j = 0;  
        while(j<i+1)  
        {  
            printf(" * ");  
            j++;  
        }  
        printf("\n");  
        i++;  
    }  
    return 0;  
}
```


do ... while Döngüsü

- **while**' dan farkı, koşulun döngünün sonunda sınanmasıdır. Yani koşula bakılmaksızın çevrime girilir, tekrarlanması koşulun doğruluğuna bağlıdır.
- Dolayısıyla ilk çevrim koşulsuz, sonrakiler koşullu yürütülür.
- İki ya da daha fazla koşulun mantıksal operatörlerle birleştirilerek verilmesi de mümkündür.




```
do  
{  
    işlem-1;  
    ....  
    işlem-N;  
} while(koşul);
```



- Programda bir kod parçası yürütüldükten sonra tekrarlanması istenip istenmediği kullanıcıya sorularak gelen yanıtı göre tekrar sağlanacak durumlarda kullanılabilir.
- Döngü sonunda while ifadesinden sonra mutlaka ; eklenmelidir.

Örnek: Klavyeden girilen iki sayıyı toplayan ve kullanıcının isteğine göre işlemi tekrarlayan bir C kodu yazınız.

```
#include<stdio.h>
main() {
int a,b; char kr;
do { printf("Bir sayi giriniz:\n");
    scanf("%d",&a);
    printf("Bir sayi daha giriniz:\n");
    scanf("%d",&b);
    printf("%d+%d=%d\n",a,b,a+b);
    printf("Baska islem yapmak istiyor musunuz? (E/H)\n");
    kr=getch();
} while(kr!='h' && kr!='H');
}
```

for Döngüsü

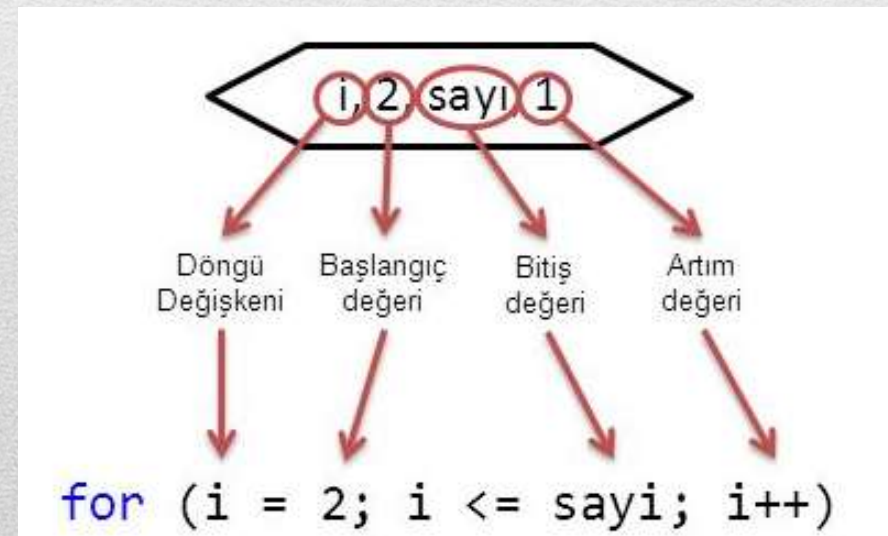
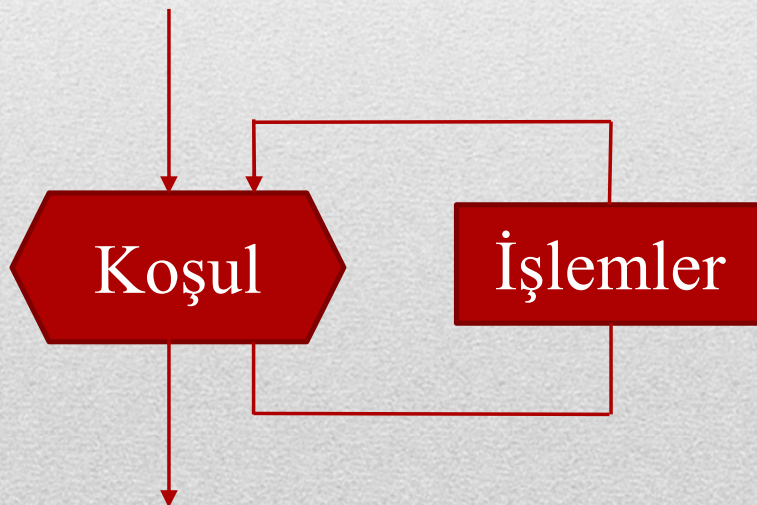
- Çevrime girmeden koşulun sınandığı bir diğer döngü deyimidir.
- **while**' dan farkı, sayacın döngü deyiminin içinde yer alması ve yine döngü deyimi içinde artırılmasıdır.

```
for (ilk_deger;koşul;artim)
{
    işlem-1;
    ....
    işlem-N;
}
```

- Parantez içindeki ifadeler ; ile ayrılmalıdır.
- for deyiminden sonra ; konmaz.

```
for (i=0; i<=15; i++)
```


for Döngüsü	while Döngüsü
<pre> for(int i = 1; i <= 5; i++) { printf("%d",i); } </pre> <p>Program çıktısı: 12345</p>	<pre> int i = 1; while(i <= 5) { printf("%d",i); i++; } </pre> <p>Program çıktısı: 12345</p>



- **for** döngüsünde noktalı virgülle ayrılan kısımlar içinde birden fazla komut varsa, bu komutların virgül ile ayrılması gerekir.

```
#include<stdio.h>
main() {
int a,b;
for (a=4, b=6; a*b<=100; a++, b+=2)
    { printf("a değişkeninin şu anki değeri: %d\n",a);
      printf("b değişkeninin şu anki değeri: %d\n",b);
      printf("Çarpma işlemi sonucu:%d\n",a*b);
    } return 0;
}
```

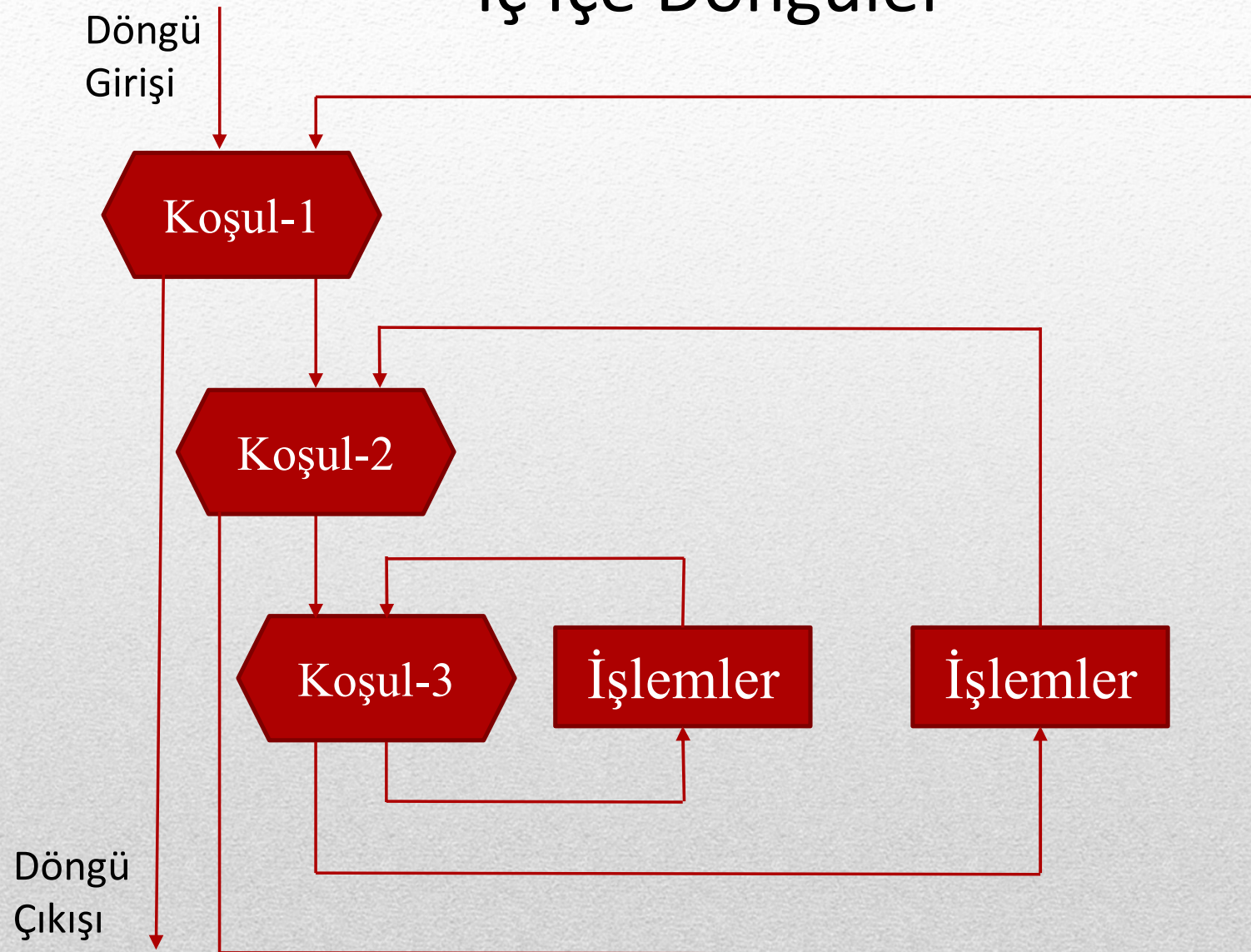

- Döngü ifadesi içinde başlangıç, koşul ve artım/azaltım parametrelerinin herhangi biri kullanılmayabilir. Ancak herhangi biri olmadığında döngünün nasıl yürütüleceği iyi anlaşılmalıdır.

```
for (; ;)  
    printf(" Sonsuz döngüye girdiniz.\n ");
```

```
...  
for(k=1; k<10;)  
    printf("%f\n ", sqrt(k++));  
...
```

```
...  
x=0;  
for(; x<5; x+=2)  
    printf("%d\n ", x);  
...
```

İç İçe Döngüler



Örnek: Çarpım tablosunu yazdıran program kodu

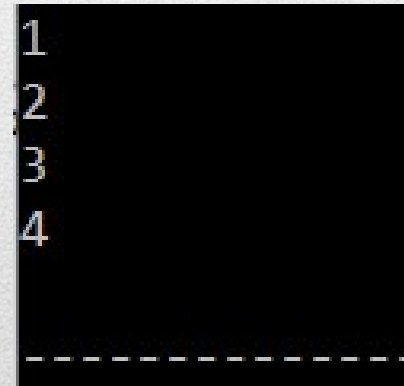
```
#include<stdio.h>
main(){
    int num,i,j;
    printf("Please enter an integer number: ");
    scanf("%d", &num);
    for(i=1; i<=num; i++){
        for(j=1; j<=10; j++)
            { printf("%d * %d = %d\t", i,j, i*j); }
        printf("\n"); }
    return(0); }
```

→ Parantez
eklenmeyebilir.

break Deyimi

Döngü içerisinde belli şartlar oluştuğunda döngüyü sonlandırmak için kullanılır.

```
#include<stdio.h>
main() {
    int i;
    for(i=1; i<7; i++)
    {    if(i==5) { break; }
        printf("%d \n", i);
    }
    return 0; }
```



1
2
3
4

continue Deyimi

Derleyici bu deyimle karşılaştığında, döngünün bulunduğu iterasyonu sonlandırır ve bir sonraki iterasyona geçer. Başka bir deyişle, döngünün o iterasyonunda continue deyiminden sonraki komutlar yürütülmez.

```
#include<stdio.h>
main() {
    int i;
    for(i=1; i<7; i++)
        {   if(i==5) { continue; }
            printf("%d \n", i);
        }
    return 0; }
```

```
1
2
3
4
6
-----
```


ÖDEV:

Sizlerle paylaşılan “c-dongu-ornek.pdf” dosyası içindeki örneklerin her birini önce cevaplara bakmadan kendiniz Dev-C++ üzerinde gerçekleştirmeye çalışınız. Daha sonra yazdığınız kodu bu dosyadaki cevaplarla karşılaştırınız. Hangisinin daha etkin olduğunu yorumlayınız.