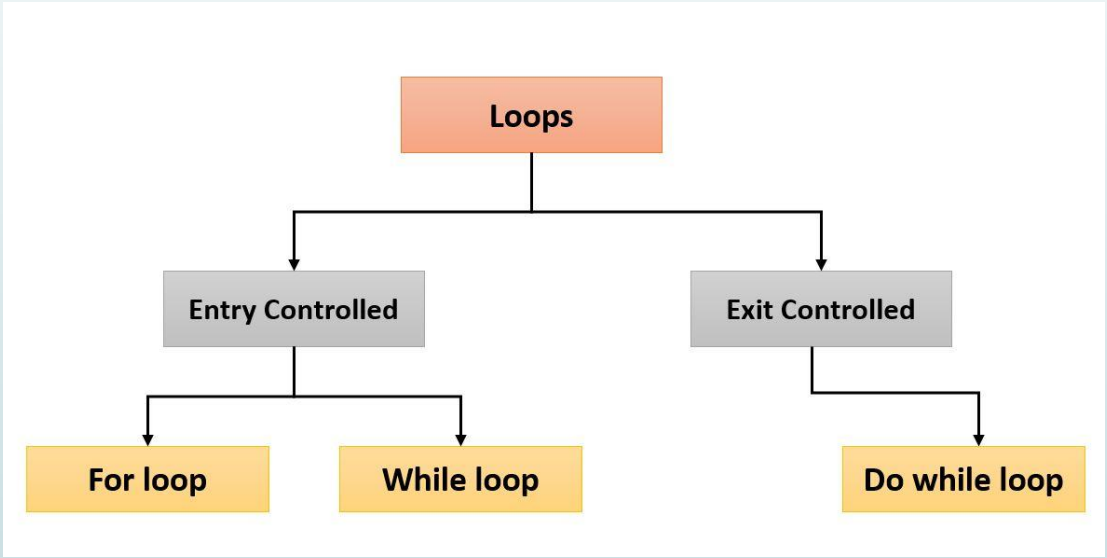
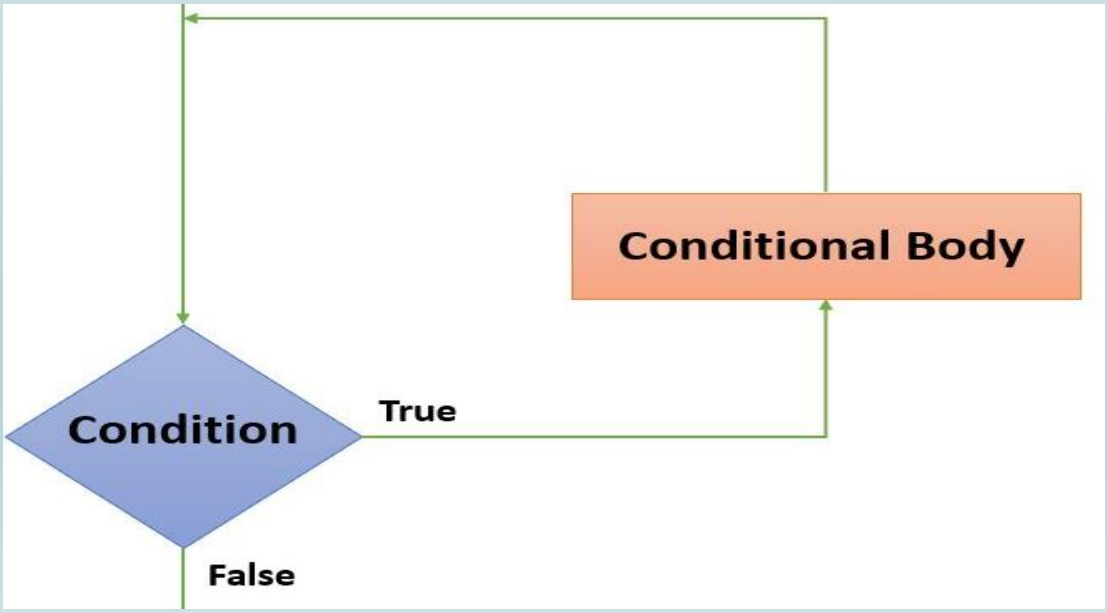
KONTROL YAPILARI-2

Hazırlayan : Muhammed Esad Goncaloğlu

DÖNGÜ YAPILARI

- Döngü, belirli bir koşul sağlanana kadar aynı işlemin birden çok kez tekrarlanması olarak tanımlanabilir.
- C dilinde kullanılan üç tür döngü vardır.
- Bunlar for, while ve do-while döngü yapılarıdır.
- Kodun yeniden kullanılabilirliğini sağlar.
- Döngüleri kullanarak aynı kodu tekrar tekrar yazmamıza gerek kalmaz.
- Döngüleri kullanarak, veri yapılarının (dizi veya bağlantılı listeler) öğeleri arasında gezinebiliriz.





FOR DÖNGÜSÜ

- C Programlamada en çok kullanılan döngüdür.
- Bu döngü, bir ifadeyi birkaç kez yinelemek için tasarlanmıştır.
- For döngüsünün üç değer alan kısmı vardır.
- Başlatma, bir for döngüsünde yalnızca bir kez gerçekleştirilir.
- Koşul, her yineleme için sabit bir değeri test eden ve onunla karşılaştıran ifadeler kümesidir. Koşul sonuçları yanlışsa yinelemeleri durdurur.
- Artırma veya eksiltme, koşul değerini ihtiyacına göre değiştirir.

```
for (initialization; condition; increment/decrement)
{
// statement to be executed
}
```

```
#include <stdio.h>
int main()
{
    int i;
    for (i=0; i<10; i++)
    {
        printf("%d ", i);
    }
    printf("for loop body finished");
    return 0;
}
```

WHILE DÖNGÜSÜ

- En temel döngü türüdür ve diğer döngü yapılarına göre daha basit bir yapısı vardır.
- Parantez bloğunun içerisindeki durumu değerlendirir.
- Önce koşulu kontrol eder ve doğruysa programın döngü gövdesinin içine girmesine izin verir ve koşul başarısız olduğunda program döngüden çıkar.
- Genelde arttırma ve eksiltme operatörleri ile beraber çalışır.

```
while (expression)
{
  // statement to be executed or repeated
}
```

```
#include <stdio.h>
int main( )
{
int n = 15;
while (n>0) {
printf("%d ",n);
n = n - 1;
}
printf("Loop body finished");
return 0;
}
```

```
15 14 13 12 10 9 8 7 6 5 4 3 2 1 Loop body finished
```

DO-WHILE DÖNGÜSÜ

- ‘while’ döngüsüne çok benzerdir.
- Tek farkı koşulun en son kontrol edilmesidir.
- Koşul döngü gövdesinin sonunda bildirildiği için bloğun içindeki ifade yanlış olsa bile en az bir kere çalıştırır.
- İşlem ‘do’ kısmında gerçekleşir. ‘while’ kısmında sadece koşul sağlaması yapılır. Koşul sağlandıkça döngü çalışır. Sağlanmadığında döngü biter.

```
do
{
// statement to be executed or repeated
}
while (condition);
```

```
#include <stdio.h>
int main()
{
int n;
do {
printf("Enter value: ");
scanf("%d", &n);
printf("The value is: %d\n",n);
} while (n != 20);
printf("do-while loop body finished");
return 0;
}
```

BREAK VE CONTINUE DEYİMLERİ

- Genellikle döngülerde ve ‘switch-case’ yapılarında kullanılır.
- ‘break’ deyimi, bir döngüden çıkmak için kullanılır. ‘switch-case’ de kullanımı ise sadece ilgili ‘case’ bölümünün çalıştırılması içindir.
- ‘continue’ deyimi ise, belirtilen bir koşul oluşursa bir yinelemeyi (döngüde) keser ve döngüdeki bir sonraki yinelemeyle devam eder.

```c
int i;

for (i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  printf("%d\n", i);
}
```

```c
int i;

for (i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  printf("%d\n", i);
}
```

İÇ İÇE DÖNGÜ YAPILARI

- Döngülerin iç içe geçmesi, ifadelerin başka bir döngü içinde döngüye alınmasına izin veren özelliktir.
- Başka bir döngü içinde herhangi bir sayıda döngü tanımlanabilir, yani herhangi bir sayıda döngü tanımlamak için herhangi bir kısıtlama yoktur.
- Genellikle diziler ile beraber kullanılır.
- Herhangi bir döngü türünü başka bir döngü içinde tanımlanabilir.

```
for (initialization; condition; update)
{
   for(initialization; condition; update)
   {
        // inner loop statements.
   }
   // outer loop statements.
}
```

```
for(int i=1;i<=n;i++)  // outer loop
{
   for(int j=1;j<=10;j++)  // inner loop
   {
      printf("%d\t",(i*j)); // printing the value.
   }
   printf("\n");
}
```

KAYNAKÇA

- https://www.javatpoint.com/c-loop
- https://usemynotes.com/what-is-loop-statement-in-c/
- https://www.w3schools.com/c/c_break_continue.php
- https://www.javatpoint.com/nested-loops-in-c