C
Kontrol İfadeleri

Dr. Öğr. Üyesi M. Ozan AKI

Rev 1.0

true (doğru) ve false (yanlış)

Kontrol ifadelerinde kullanılan koşullar mantıksal durumları ifade eder.

Mantıksal bir durum sadece **doğru** ya da **yanlış** olabilir.

C dilinde doğru ve yanlış mantıksal durumları için özel bir veri tipi olmayıp, yanlış mantıksal durumu için 0 (sıfır), doğru mantıksal durumu için sıfırdan farklı bir değer kullanılır.

Kontrol İfadeleri

Belirli şartlar altında program akışını değiştiren ifadeleridir.

Kontrol İfadeleri

Şart İfadeleri

- if else
- switch case

Döngüler

- for
- while
- do while

if else

```
if(koşul)
{
…
}
else
{
…
}
```

Belirli bir şarta bağlı olarak bir ifade ya da ifade bloğunun çalışmasını yada çalışmamasını sağlar.

Belirtilen **koşul** doğru (true) ise, if parantezini takip eden ifade ya da blok, koşul doğru değilse (false) **else** ifadesi ya da bloğu çalışır.

else kullanmak seçimliktir.

if else

```
if(a>0)
  if(b<5)
    if(c==3)
      d=a+b;
    else
      d=a-c;
  else
      d=a*b;
```

Blok belirtilmeden iç içe kullanılan if ifadelerinde kullanılan else;

else kullanmayan en yakın if ifadesine ait olur.

Belirsizlik ya da şüphe duyduğunuz anda { } blok ifadelerini kullanın.

if else

if(koşul)
…
else if(koşul)
…
else if(koşul)
…
else if(koşul)
…
else
…

Bazı durumlarda birden fazla şartın ardaşık olarak karşılaştırılması gerekebilir.

Bu gibi durumlarda if else ifadeleri kullanılarak her koşul ayrı ayrı kontrol edilir.

Herhangi koşul doğru olduğunda, o koşula ait ifade çalışır ve tüm if else bloğu sonlandırılır.

Eğer hiçbir koşul doğru olmadıysa ifadenin en sonunda else ifadesi çalışır. Bu ifade seçimliktir.

switch case default

```
switch(değişken)
{
case değer1:
…
break;
case değer2:
…
break;
case değer3:
…
break;
default: …
}
```

Ard arda kullanılan if else ifadelerine benzer bir şekilde, bir değişkenin değerini çok sayıda değişken ile karşılaştıran bir ifadedir.

değişkenin değeri her bir **case değeri** ile karşılaştırılır. Hangisi ile eşleşirse, program akışı o case ifadesineden devam eder.

switch case default

```
switch(değişken)
{
case değer1:
...
break;
case değer2:
...
break;
case değer3:
...
break;
default: ...
}
```

Eğer hiçbir **case** değeri eşleşmedi ise, **default** ifadesi çalışır.

default ifadesi seçimliktir. Eğer hiçbir case değeri eşleşmediği halde default ifadesi de kullanılmadıysa switch ifadesinde hiçbir komut çalıştırılmamış olur.

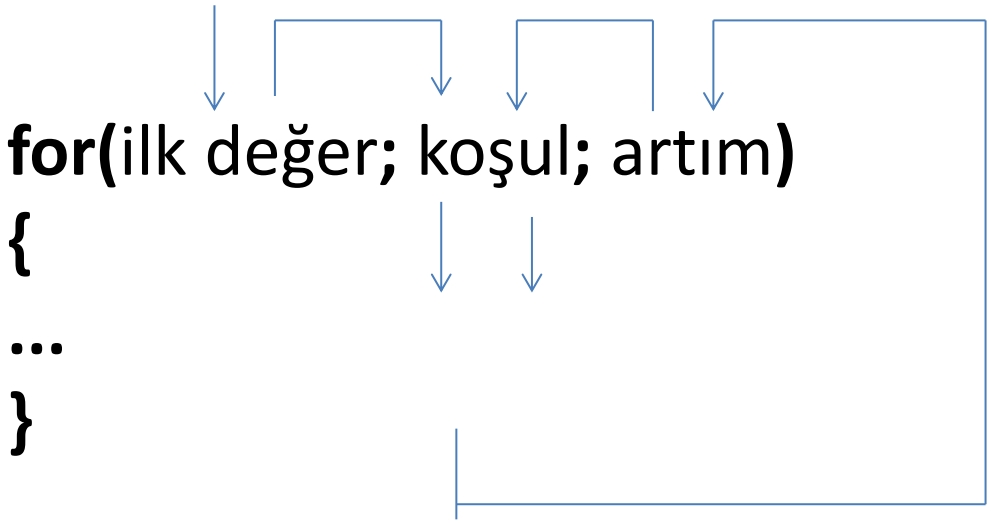
switch case default

```
switch(değişken)
{
case değer1:
…
break;
case değer2:
…
break;
case değer3:
…
break;
default: …
}
```

Program akışı, eşleşen herhangi case ifadesine girdiğinde, ardaşık gelen tüm case ifadeleri çalıştırılır.

Eğer bir case ifadesi eşleştikten sonra, ardından gelen case ifadelerinin çalışması istenmiyorsa case bloğunun sonuna **break** konularak switch ifadesinin sonlanması sağlanır.

for



Genellikle, sayısı spesifik olarak belirli döngüler oluşturmak için kullanılır.

for bloğuna ilk girişte *ilk değer* çalıştırılır. ardından *koşul* kontrol edilir. koşul doğru ise if bloğu çalıştırılır. Bloktan çıkıldığında önce artım ifadesi çalıştırılır ve ardından tekrar koşul kontrol edilir.

Koşul doğru olduğu sürece for bloğu tekrar tekrar çalıştırılır.

for

```
int i;
for(i=0; i<100; i++)
{
...
}
```

Tipik bir for bloğu bu şekilde olabilir. for bloğu içerisindeki komutlar, for ifadesi tamamlandığında toplam 100 defa çalıştırılmış olur.

for

```
int i=0;
for(;i<100;)
{
i++;
...
}
```

for bloğu içerisindeki ifadelerin tümü seçimliktir. Ancak kullanılmayan ifade olsa dahi noktalı virgüller mutlaka konulmalıdır.

Eğer koşul ifadesi belirtilmemiş ise doğru (true) sayılır ve sonsuz döngü meydana gelir.

for(;;) { ... } → sonsuz döngüdür.

while

while(*koşul*)

{

…

}

while döngüsünde, **koşul** doğru olduğu sürece while bloğu içerisindeki komutlar çalıştırılır.

Eğer ilk anda koşul doğru değilse while bloğuna hiç girilmez.

do while

do {

...

} while(*koşul***)**

do while döngüsünde de, **koşul** doğru olduğu sürece do while bloğu içerisindeki komutlar çalıştırılır.

while döngüsünden temel farkı, koşul kontrol edilmeden önce mutlaka en az bir defa döngü bloğu içerisindeki komutların çalıştırılmasıdır.

Neden? belki koşul, döngü içerisindeki komutların çalıştırılması sonucu ortaya çıkıyor olabilir.

break ve continue

break;

içerisinde bulunduğu bloğu hemen sonlandırarak, bloktan çıkılmasını sağlar. Kendisinden sonraki komutlar çalıştırılmaz.

for, while, do while, ve switch case kontrol ifade bloklarında kullanılır.

break ve continue

continue;

İçerisinde bulunduğu döngü bloğunun o anda kalan komutların çalıştırmayıp hemen bir sonraki tekrarın başlatılmasını sağlar.

Koşul yanlış olmuşsa döngüden çıkılır, koşul halen doğru ise döngüye devam edilir.

for, while, do while, döngü bloklarında kullanılır.

{ bloklar }

{ ve } parantezleri içerisinde belirtilen tüm ifadeler mantıksal olarak bir bütün oluşturur.

Genellikle bloklar, fonksiyon ve kontrol ifadelerinin gövdelerini oluşturmak amacıyla kullanılır.

Eğer bir kontrol ifadesi sadece bir c ifade satırından oluşuyorsa gerçekte blok kullanmak zorunlu değildir.

{ bloklar }

```
if(a<b)
{
   printf(“a kucuk b\n”);
}

if(a<b) printf(“a kucuk b\n”);
```