C Programlama Dili

Dr. Öğr. Üye. Zafer Civelek

Tekrarlamanın Temelleri

Çoğu program tekrarlama veya döngü içerir. Bir döngü, döngü devam koşulu doğru kaldığı sürece bilgisayarın tekrar tekrar yürüttüğü bir talimatlar grubudur. Tekrarlamanın iki anlamından bahsedilmişti;

1. sayaç kontrollü tekrarlama

2. sonlandırma kontrollü tekrarlama

Sayaç kontrollü tekrarlamada, tekrarlama sayısını saymak için bir kontrol değişkeni kullanılır. Kontrol değişkeni talimatlar grubunun yapıldığı her sefer (genelde 1) arttırılır. Kontrol değişkeninin değeri doğru sayıda tekrarlamanın yapıldığını gösterdiğinde, döngü sonlanır ve yürütme, tekrarlama ifadesinden sonraki ifade ile devam eder.

Sayaç Kontrollü Tekrarlama

sonlandırma kontrollü tekrarlama ise

1. Kesin tekrarlama sayısı önceden bilinmediği ve,

2. Döngü, döngünün yapıldığı her sefer veri alan ifadeler içerdiği zaman

tekrarlamayı kontrol etmek için sonlandırma değerleri kullanılır. Sonlandırma değeri «veri sonuna» işaret eder. Sonlandırmalar normal veriden farklı olmak zorundadır. Sayaç kontrollü tekrarlama,

1. kontrol değişkeninin (veya döngü sayacı) ismi.

2. kontrol değişkeninin başlangıç değeri.

3. Her döngü seferinde kontrol değişkeninin değiştirileceği arttırma (veya azaltma)

4. kontrol değişkeninin son değeri (yani döngünün devam edip etmeyeceğini) karşılaştıran koşul.

maddelerini gerektirir.

Sayaç Kontrollü Tekrarlama

1'den 10'a kadar sayıları yazan basit programı göz önüne alalım.

```
1   //sayaç kontrollü tekrarlama
2   #include<stdio.h>
3   //main fonksiyonu program yürütmesine başlar
4   int main(void){
5       unsigned int counter=1;//başlangıç değeri verme
6       while(counter<=10){//tekrarlama koşulu
7           printf("%u\n",counter);//sayacı göster
8           ++counter;//arttırma
9       }//while sonu
10  }//main fonksiyon sonu
```

```
1
2
3
4
5
6
7
8
9
10
```

Sayaç Kontrollü Tekrarlama

counter'ın tanımlaması ve başlangıç değeri verilmesi aynı zamanda;

unsigned int counter;

counter=1;

gibi de yazılabilirdi. Tanımlama yürütülebilir değildir, ancak atama öyledir.

++counter;//arttırma

ifadesi, döngünün yapıldığı her sefer döngü sayacını bir arttırır. while ifadesindeki döngü devam koşulu kontrol değişkeni değerinin 10'dan küçük veya eşit olup olmadığını kontrol eder.

Sayaç Kontrollü Tekrarlama

Bu while'ın gövdesi, kontrol değişkeni 10 olduğu zaman bile yapılır. Kontrol değişkeni 10'u geçtiği zaman döngü sonlanır. Sayaca başlangıç değeri olarak sıfır vererek, while ifadesini

while(++counter<=10)

printf("%u\n",counter);

ile değiştirerek program daha kısa yapılabilir.

Yaygın programlama hatası 4.1

Ondalık değerler yaklaşık olabilir, bu nedenle sayan döngüleri ondalık değerler ile kontrol etmek tam doğru olmayan sayaç değerleri ve sonlandırma karşılaştırmaları ile sonuçlanabilir.

Sayaç Kontrollü Tekrarlama

Hata önleme önerisi 4.1

sayan döngüleri tam sayı değerler ile kontrol edin.

İyi programlama uygulaması 4.1

çok sayıda kümeleme seviyesi bir programın anlaşılmasını zorlaştırabilir. Kural olarak, üç kümeleme seviyesinden fazlasını kullanmamaya çalışın.

İyi programlama uygulaması 4.2

kontrol ifadelerinden önce ve sonra dikey boşluk bırakma ve kontrol ifadesi başlıkları içerisinde kontrol ifadelerinin gövdesinde boşluk bırakma, programlarda okunabilirliği mükemmel şekilde iyileştiren iki boyutlu bir görünüm kazandırır.

for Tekrarlama İfadesi

for tekrarlama ifadesi

for tekrarlama ifadesi, sayaç kontrollü tekrarlamanın tüm detaylarını sağlar. 1'den 10'a kadar sayıları yazdıran programı tekrar for komutu ile yazalım.

```c
/* Fig. 4.2: fig04_02.c
   Counter-controlled repetition with the for statement */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   int counter; /* define counter */

   /* initialization, repetition condition, and increment
      are all included in the for statement header. */
   for ( counter = 1; counter <= 10; counter++ ) {
      printf( "%d\n", counter );
   } /* end for */

   return 0; /* indicate program ended successfully */
} /* end function main */
```

```
1
2
3
4
5
6
7
8
9
10
```

for Tekrarlama İfadesi

```c
/* Fig. 4.2: fig04_02.c
   Counter-controlled repetition with the for statement */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   int counter; /* define counter */

   /* initialization, repetition condition, and increment
      are all included in the for statement header. */
   for ( counter = 1; counter <= 10; counter++ ) {
      printf( "%d\n", counter );
   } /* end for */

   return 0; /* indicate program ended successfully */
} /* end function main */
```

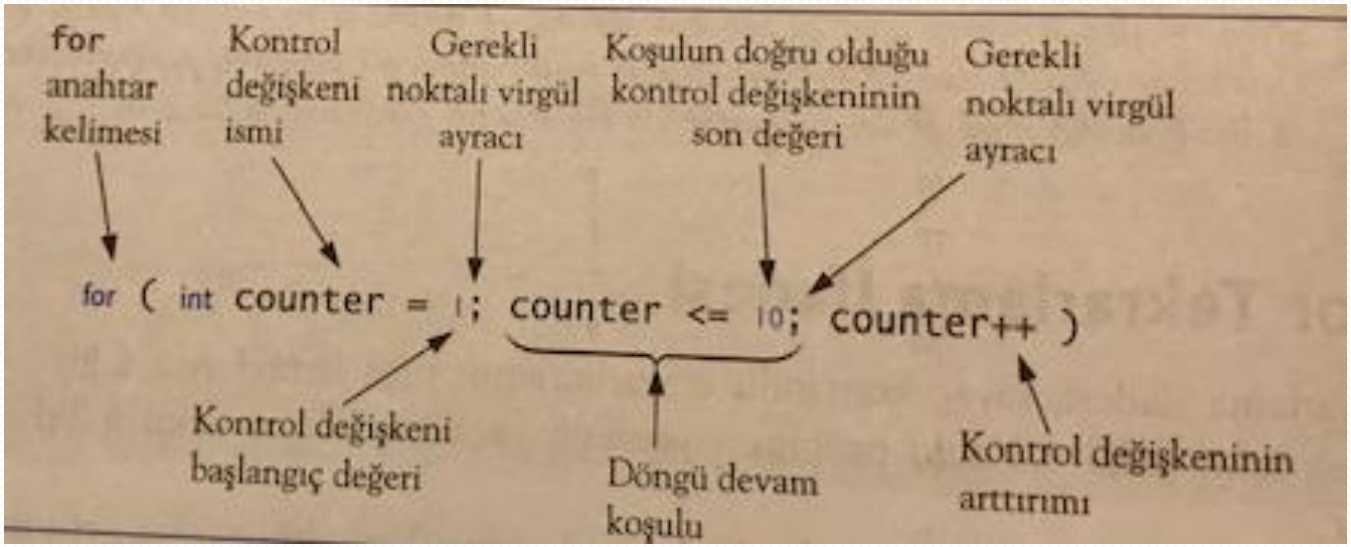
for Tekrarlama İfadesi

Program şu şekilde çalışır; for ifadesi çalışmaya başladığı zaman, counter kontrol değişkenine başlangıç değeri olarak 1 verilir. Sonra counter <=10 şartı kontrol edilir. counter'ın başlangıç değeri 1 olduğu için koşul sağlanır, böylece printf ifadesi counter değerini, yani 1'i yazar. counter kontrol değişkeni daha sonra ++counter ifadesi tarafından bir arttırılır ve döngü devam karşılaştırması ile tekrar başlar. Kontrol değişkeni şimdi 2 olduğundan, son değer aşılmamıştır bu nedenle program printf ifadesini tekrar eder. Bu süreç counter kontrol ifadesi 11 olana kadar devam eder. 11 olduğunda döngü devam testi yanlış yani 0 değerini alır böylece döngü sona erer. Program for ifadesinden sonraki ilk ifadeyi icra ederek devam eder.

for Tekrarlama İfadesi

for ifadesi başlık öğeleri

for ifadesinin gövdesinde birden fazla ifade varsa, döngü gövdesini tanımlamak için küme parantezleri gereklidir. C standardı, for başlığının başlangıç kısmında kontrol değişkenini tanımlamamıza (int counter=1) izin verir. Bu özellik bazı derleyicilerde bulunmaz.



for Tekrarlama İfadesi

counter<=10 döngü devam koşulunda eğer yanlışlıkla counter<10 yazılsaydı, döngü 9 kere yürütülecekti. Bu bir tekrarı unutma hatası denilen yaygın bir mantık hatasıdır.

Hata önleme önerisi 4.2

Bir while veya for ifadesinde son değer kullanma ve <= ilişkisel işlemini kullanma bir tekrarı unutma hatalarını önlemeye yardımcı olabilir. 1'den 10'a kadar değerleri yazmak için kullanılan bir döngü için, mesela döngü devam koşulu counter<11 veya counter<10 yerine counter<=10 olmalıdır.

for Tekrarlama İfadesi

for(ifade1;ifade2;ifade3){
ifade
}

for ifadesinin genel biçimi, ifade1'in döngü kontrol değişkenine başlangıç değeri verildiği, ifade2'nin döngü devam koşulu olduğu, ve ifade3'ün, kontrol değişkenini arttırdığı şeklindedir. Çoğu durumda, for ifadesi, bir while ifadesi karşılığı ile şu şekildedir;

İfade1;
while(ifade2){
ifade
ifade3;
}

for Tekrarlama İfadesi

for ifadesinde üç ifade kullanımı isteğe bağlıdır. Eğer ifade2 ihmal edilirse, C, koşulun doğru olduğunu kabul eder, bu da sonsuz döngü oluşturur. Programın başka bir yerinde kontrol değişkenine başlangıç değeri verildiyse, ifade1 ihmal edilebilir. Arttırma, for ifadesinin gövdesindeki ifadeler tarafından hesaplanıyorsa veya hiçbir arttırmaya ihtiyaç yoksa ifade3 ihmal edilebilir. for ifadesi içindeki arttırma ifadesi, for gövdesinin sonundaki bağımsız bir C ifadesi gibi davranır. Bu nedenle,

counter=counter+1;

counter+=1;

++counter;

counter++;

ifadelerinin hepsi for ifadesinin arttırma kısmına denktir.

for Tekrarlama İfadesi

Genelde *counter*++ ifadesi programcılar tarafından tercih edilmektedir.

Yaygın programlama hatası 4.2

Bir for başlığında noktalı virgül yerine virgül kullanmak bir söz dizim hatasıdır.

Yaygın programlama hatası 4.3

Bir for başlığının hemen sağına noktalı virgül koymak o for ifadesinin gövdesini boş bir ifade yapar. Bu normal olarak mantıksal bir hatadır.

for Tekrarlama İfadesi

for ifadesi ile ilgili notlar

1. Başlangıç değeri verme, döngü devam koşulu ve arttırma aritmetik işlemler içerebilir. mesela x=2 ve y=10 ise

for(j=x;j<=4*x*y;j+=y/x) ifadesi

for (j=2;j<=80;j+=5) ifadesine denktir.

2. Arttırma negatif olabilir.

3. Döngü devam koşulu başlangıçta yanlış ise döngü gövdesi yürütülmez. Bunun yerine yürütme for ifadesini takip eden ifade ile devam eder.

for Tekrarlama İfadesi

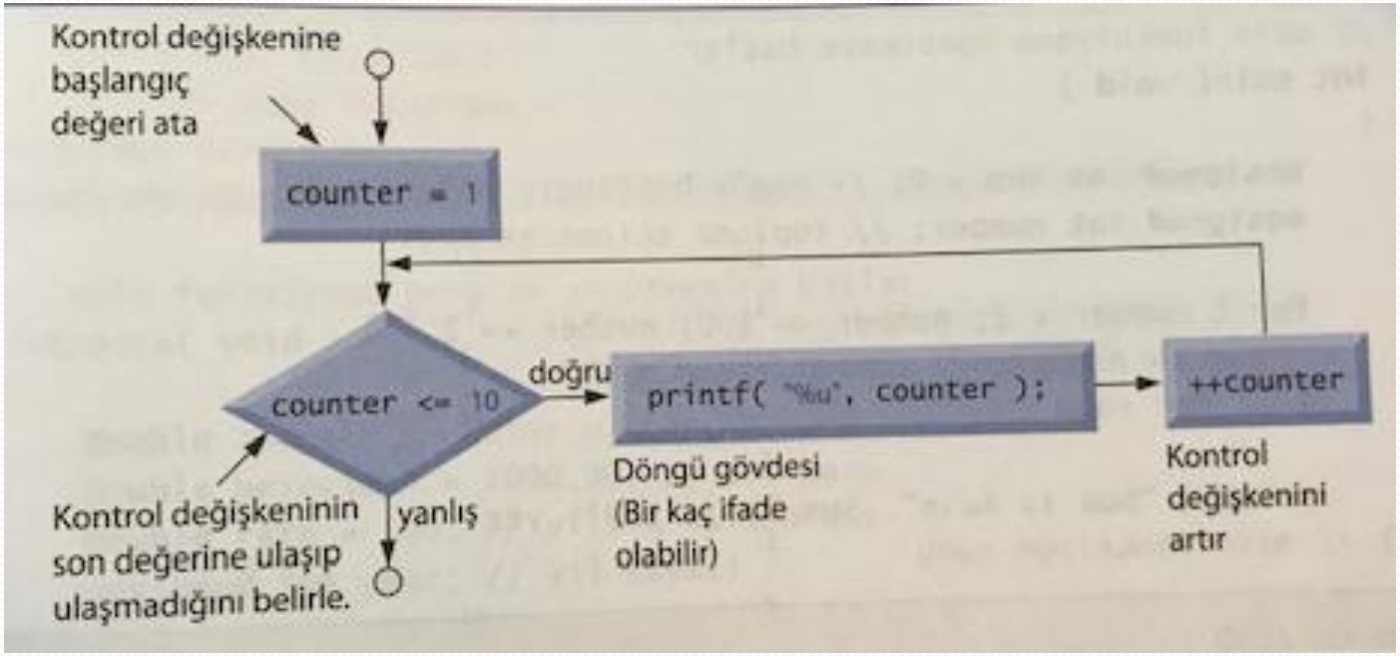
4. Kontrol değişkeni çoğunlukla kullanılır ama olması gerekmez. Döngü gövdesi içerisinde hiç bahsetmeden tekrarlamayı kontrol etmek için kontrol değişkeni kullanmak yaygındır.

5. for ifadesi de while ifadesi gibi akış diyagramı halinde gösterilebilir. mesela

```
int counter;
for (counter=1;counter<=10;++counter)
    printf("%u",counter);
```

ifadesinin akış diyagramı şu şekilde gösterilebilir;

for Tekrarlama İfadesi



Hata önleme önerisi 4.3

Kontrol değişkeninin değeri for döngüsü gövdesi içinde değiştirilebilse bile, bu durum ince hatalara yol açar. En iyisi değiştirmemektir.

for Tekrarlama İfadesi

for ifadesi kullanılan örnekler

1. kontrol değişkenini 1'den 100'e kadar 1 arttırarak değiştirme

```
for (i=1;i<=100;++i)
```

2. kontrol değişkenini 100'den 1'e -1 arttırarak (1 azaltarak) değiştirme

```
for (i=100;i>=1;--i)
```

3. kontrol değişkenini 7'den 77'ye 7 şer değiştirmek

```
for (i=7;i<=77;i+=7)
```

for Tekrarlama İfadesi

4. kontrol değişkenini 20'den 2'ye -2 şer değiştirmek

for (i=20;i>=2;i-=2)

5. kontrol değerlerini şu şekilde değiştirin: 2, 5, 8, 11, 14, 17.

for (i=2;i<=17;i+=3)

6. kontrol değişkenini şu şekilde değiştirin: 44, 33, 22, 11, 0.

for (i=44;i>=0;j-=11)

Örnek: 2'den 100'e kadar olan tüm çift sayıların toplamını bulan programı for ifadesi ile yapınız.

for Tekrarlama İfadesi

```c
//for ile toplama
#include<stdio.h>

//main fonksiyonu yürütmeye başlar
int main (void){

    unsigned int sum=0;//sum'a başlangıç değeri ver
    unsigned int number;//toplama eklenecek sayı

    for (number=2;number<=100;number+=2){
        sum+=number;//number'a sum'ı ekle
    }//for sonu

    printf("Toplam=%u\n",sum);//sum'ı yaz
}//main fonksiyonu sonu
```

```
Toplam=2550
```

for Tekrarlama İfadesi

for ifadesinin gövdesi, virgül işlemi kullanarak şu şekilde olabilir;

```
for (number=2;number<=100;sum+=number,number+=2){
}
```

sum=0 başlangıç değeri verme de for'un başlangıç kısmına yerleşebilir.

```
for (number=2,sum=0;number<=100;sum+=number,number+=2){
}
```

switch çoklu seçim ifadesi

İyi programlama uygulaması 4.3

for gövdesindeki ifadeler for başlığına yerleştirilebilse de, bunu yapmaktan sakınmak lazımdır, çünkü bu yazılım program okumasını zorlaştırır.

İyi programlama uygulaması 4.4

Eğer mümkünse kontrol ifadesi satırını tek bir satırla sınırlayın.

switch çoklu seçim ifadesi

Bir algoritma nadiren de olsa kabul edilen sabit tam sayı değerlerin her biri için bir değişkenin veya ifadenin karşılaştırıldığı ve farklı eylemlerin yapıldığı bir dizi kararlar içerecektir.

switch çoklu seçim ifadesi

Buna çoklu seçim denir. C, böyle kararları vermek için switch çoklu seçim ifadesine sahiptir. switch ifadesi, bir dizi case etiketinden, isteğe bağlı default durumundan ve her durum için yürütülecek ifadelerden meydana gelir. Örnek: Bir sınavda öğrencilerin aldıkları harf notlarını sayacak bir programı switch-case komutu ile yapın.

```c
//switch ile harf notlarını sayma
#include<stdio.h>

//main fonksiyonu yürütmeye başlar
int main (void){

    int grade;//bir not
    unsigned int aCount=0;//A'ların sayısı
    unsigned int bCount=0;//B'lerin sayısı
    unsigned int cCount=0;//C'lerin sayısı
    unsigned int dCount=0;//D'lerin sayısı
    unsigned int fCount=0;//F'lerin sayısı

    puts("Harf notunu girin.");
    puts("Girisi sonlandirmak icin EOF karakterini girin.");

    //kullanıcı dosya sonu anahtar dizini girene kadar dön
    while((grade=getchar())!=EOF){

        //hangi notun girildigine karar ver
        switch(grade){//while içine kümelenmiş switch

```

```
23          case'A'://not büyük A
24          case'a'://veya küçük a
25          ++aCount;//aCount'ı arttır
26          break;//switch den çıkmak için zorunlu
27
28          case'B'://not büyük B
29          case'b'://veya küçük b
30          ++bCount;//bCount'ı arttır
31          break;//switch den çıkmak için zorunlu
32
33          case'C'://not büyük C
34          case'c'://veya küçük c
35          ++cCount;//cCount'ı arttır
36          break;//switch den çıkmak için zorunlu
37
38          case'D'://not büyük D
39          case'd'://veya küçük d
40          ++dCount;//dCount'ı arttır
41          break;//switch den çıkmak için zorunlu
42
43          case'F'://not büyük F
44          case'f'://veya küçük f
45          ++fCount;//fCount'ı arttır
46          break;//switch den çıkmak için zorunlu
```

```
47
48          case'\n'://girilen yeni satırları
49          case'\t'://sekmeleri
50          case' '://ve boşlukları gözardı et
51          break;//switch den çık
52
53          default://diğer tüm karakterleri yakala
54          printf("Yanlis harf girdiniz.");
55          puts("Yeni bir harf giriniz");
56          break;//isteğe bağlı
57          }//switch sonu
58      }//while sonu
59
60      //sonuçların özet çıktısı
61      puts("\nHerbir harf notunun toplami");
62      printf("A: %u\n",aCount);//A notlarının sayısını gösterir
63      printf("B: %u\n",bCount);//B notlarının sayısını gösterir
64      printf("C: %u\n",cCount);//C notlarının sayısını gösterir
65      printf("D: %u\n",dCount);//D notlarının sayısını gösterir
66      printf("F: %u\n",fCount);//F notlarının sayısını gösterir
67
68  }//main fonksiyonu sonu
69
```

```
Harf notunu girin.
Girisi sonlandirmak icin EOF karakterini girin.
a
b
c
C
A
d
f
C
E
Yanlis harf girdiniz.Yeni bir harf giriniz
D
A
b
^Z

Herbir harf notunun toplami
A: 3
B: 2
C: 3
D: 2
F: 1
```

switch çoklu seçim ifadesi

Bu programda, kullanıcı bir sınıfın harf notlarını girer. while başlığında;

while((grade=getchar()!=EOF))

parantez içindeki (grade=getchar()) ifadesi önce yürütülür. getchar fonksiyonu (<stdio.h>'den) klavyeden bir karakter okur ve bu karakteri grade tamsayı değişkeninde saklar. Karakterler normalde char türü değişkende saklanır. Bununla birlikte, C'nin önemli bir özelliği, karakterler bilgisayarda bir bayt tam sayı olarak temsil edildiği için karakterler herhangi bir tam sayı veri türünde saklanabilir. Bu nedenle, bir karaktere, kullanımına bağlı olarak ya bir tam sayı yada bir karakter olarak davranabiliriz.

switch çoklu seçim ifadesi

Mesela;

```
1   #include<stdio.h>
2   int main(void){
3       printf("(%c) karakterinin degeri %d dir.\n",'a','a');
4   }
```

ifadesi, a karakterini ve tam sayı değerini yazmak için %c ve %d çevrim belirteçlerini kullanır. Sonuç şu şekildedir;

```
(a) karakterinin degeri 97 dir.
```

97 tam sayısı, a karakterinin bilgisayardaki sayısal temsilidir. Bir çok bilgisayar bugün, 97'nin a harfini temsil ettiği ASCII karakter setini kullanır. Karakterler, scanf ile %c çevrim belirtecini kullanarak okunabilir.

switch çoklu seçim ifadesi

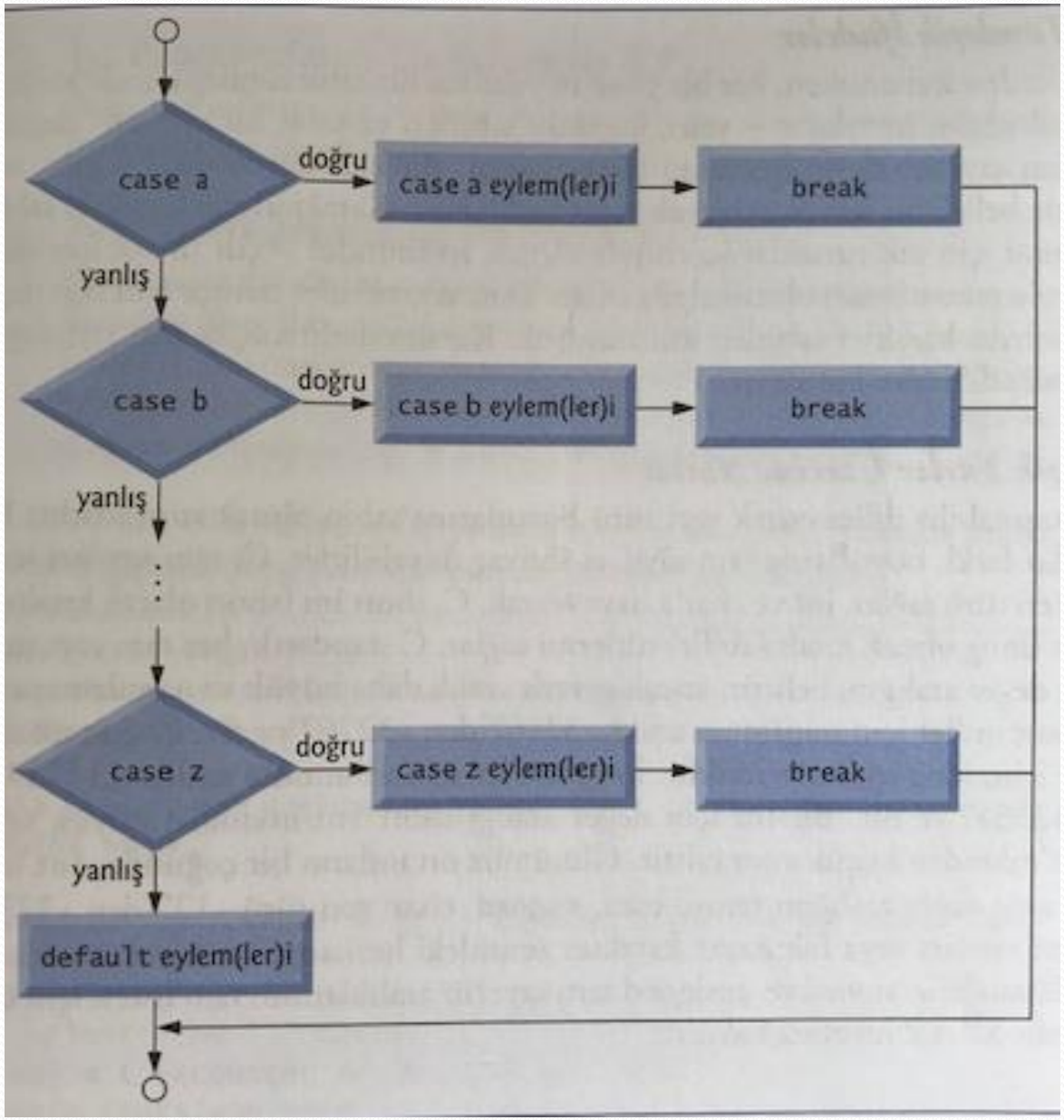
Programda, grade=getchar() atamasının değeri EOF (dosya sonu) değeri ile karşılaştırılıyor. Sonlandırma değeri olarak EOF kullanılmıştır. Bu bilgisayarda <Ctrl>Z ile girilir.

switch anahtar kelimesi, parantez içerisindeki grade değişken ismi ile takip edilir. Buna kontrol ifadesi denilir. Bu ifadenin değeri case etiketlerinin her biri ile karşılaştırılır. Mesela kullanıcının not olarak C harfini girdiğini kabul edin, C, otomatik olarak switch'deki her case ile karşılaştırılır. Eğer bir eşleşme olursa (case 'C'), o case'in ifadeleri yürütülür. C harfi durumunda cCount bir arttırılır ve break ifadesi ile switch yapısından çıkılır.

switch çoklu seçim ifadesi

break ifadesi, program kontrolünün switch ifadesinden sonraki ilk ifade ile devam etmesine neden olur. Eğer hiçbir eşleşme olmazsa default durumu yürütülür.

Her bir case bir veya daha fazla eylem içerebilir. switch'in bir case'indeki çoklu eylemler etrafında küme parantezlerine gerek olmaması, switch ifadesini diğer tüm kontrol ifadelerinden ayıran bir özelliktir. case'lerinde break olan bir switch ifadesinin akış diyagramı şu şekildedir:


case a
doğru
case a eylem(ler)i
break
yanlış
case b
doğru
case b eylem(ler)i
break
yanlış
case z
doğru
case z eylem(ler)i
break
yanlış
default eylem(ler)i

switch çoklu seçim ifadesi

Yaygın programlama hatası 4.4

Bir switch ifadesinde ihtiyaç duyulduğu halde break ifadesini unutmak bir mantık hatasıdır.

switch ifadesindeki;

```
case'\n'://girilen yeni satırları
case'\t'://sekmeleri
case' ':ve boşlukları gözardı et
break;//switch den çık
```

satırları, programın yeni satırları, sekme ve boş karakterleri atlamasına neden olur.

switch çoklu seçim ifadesi

Her seferinde bir karakter okuma bazı problemlere neden olabilir. Programa karakterleri okutmak için, Enter tuşuna basarak onları bilgisayara göndermek zorundayız. Bu, işlemek istediğimiz karakterden sonra girişte yeni bir satır yerleşmesine neden olur. Programı doğru bir şekilde çalıştırmak için bu yeni satır karakteri özel olarak işlenmek zorundadır. switch ifadesine bu case'leri ekleyerek, girişte yeni bir satır, sekme veya boşlukla karşılaşıldığında default durumundaki hata mesajının yazılmasını önlemiş oluruz. Birkaç case etiketini birlikte listelemek, her iki durumdan biri için aynı eylem kümesinin yapılacağını gösterir.

do…while

do…while tekrarlama ifadesi

do…while tekrarlama ifadesi while ifadesine benzerdir. while ifadesinde, döngü devam koşulu, döngü gövdesi icra edilmeden önce döngü başında kontrol edilir. do…while ifadesi döngü devam koşulunu döngü gövdesi yapıldıktan sonra kontrol eder. Bundan dolayı döngü gövdesi en az bir kere yürütülecektir. Bir do…while sonlandığı zaman, yürütme while kelimesinden sonraki ifade ile devam eder. Gövde içerisinde sadece bir ifade varsa küme parantezini kullanmak zorunlu değildir ama kullanılması tavsiye edilir.

do...while

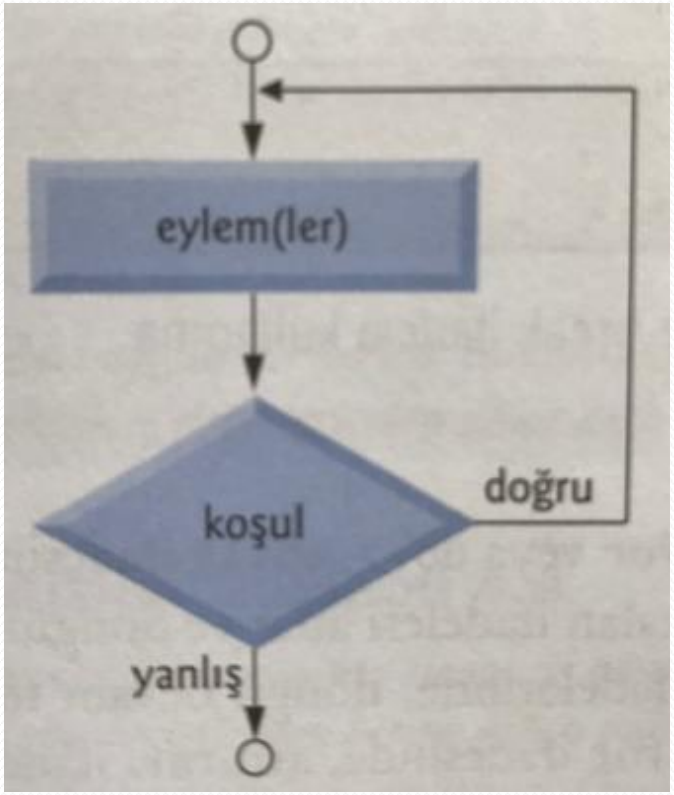
1'den 10'a kadar sayıları yazmak için do...while ifadesi kullanımı örneği;

```
1   //do...while tekrarlama ifadesi
2   #include<stdio.h>
3
4   //main fonksiyonu yürütmeye başlar
5   int main(void){
6       unsigned int counter=1;//counter'a başlangıç değeri ver
7
8       do{
9           printf("%u ",counter);//counter'ı göster
10      }while(++counter<=10);//do...while sonu
11  }//main fonksiyonu sonu
```

```
1 2 3 4 5 6 7 8 9 10
```

do…while

do…while ifadesi akış diyagramı:



break

break ifadesi

break ifadesi, bir while, for, do...while veya switch ifadeleri yürütüldüğünde ifadeden çıkışa sebeb olur. Programın yürütülmesi bir sonraki ifade ile devam eder. break ifadesinin yaygın kullanımı, bir döngüden erken çıkmak veya bir switch ifadesinin geri kalan kısımlarını atlamak içindir.

Döngü sayaç değeri 5'e geldiğinde döngüyü kıran bir program örneği:

break

```c
//for döngüsünde break ifadesi kullanma
#include<stdio.h>

//main fonksiyonu yürütmeye başlar
int main(void){

    unsigned int x;//sayaç

    for(x=1;x<=10;++x){//10 kere dön

    if(x==5){//x=5 ise döngüyü sonlandır
    printf("\ndongu x=%u degerinde sonlandirilmistir.\n",x);
        break;
    }//if sonu

    printf("%u ",x);//x'in değerini yazar
    }//for sonu

}//main fonksiyonu sonu
```

```
1 2 3 4
dongu x=5 degerinde sonlandirilmistir.
```

continue

continue ifadesi

continue ifadesi, bir while, for, do…while veya switch ifadeleri yürütüldüğünde, kontrol ifadesi gövdesindeki geri kalan ifadeleri atlar ve döngünün sonraki adımından devam eder. while ve do…while ifadelerinde döngü devam testi, continue ifadesi yürütüldükten hemen sonra yapılır. for ifadesinde, arttırma ifadesi yürütülür, sonra devam testi yapılır.

Döngü sayaç değeri 5'e geldiğinde yürütmeyi atlayıp diğer sayaç değerlerinden devam eden bir program örneği:

continue

```c
//for döngüsünde continue ifadesi kullanma
#include<stdio.h>

//main fonksiyonu yürütmeye başlar
int main(void){

    unsigned int x;//sayaç

    for(x=1;x<=10;++x){//10 kere dön

    if(x==5){//x=5 ise döngüyü atla
    printf("\ndongu x=%u degerini atlar.\n",x);
        continue;
    }//if sonu

    printf("%u ",x);//x'in değerini yazar
    }//for sonu

}//main fonksiyonu sonu
```

```
1 2 3 4
dongu x=5 degerini atlar.
6 7 8 9 10
--------------------------
```

mantıksal VE

C, basit koşulları birleştirerek daha karmaşık koşullar oluşturmak için kullanılabilen mantıksal işlemler sağlar. Mantıksal işlemler, && (mantıksal VE), || (mantıksal VEYA), ve ! (mantıksal DEĞİL) dir.

mantıksal VE (&&) işlemi

ifade1	ifade2	ifade1 && ifade2
0	0	0
0	sıfırdan farklı	0
sıfırdan farklı	0	0
sıfırdan farklı	sıfırdan farklı	1

mantıksal VE

şekildeki tablo mantıksal VE işlemini özetlemektedir. Tablo ifade1 ve ifade2'nin bütün kombinasyonlarını gösterir. C, doğru bir değeri 1'e eşitlemesine rağmen, sıfırdan farklı herhangi bir değeri doğru olarak kabul eder.

```
if(gender==1 && age>=65)
  ++seniorFemales;
```

ifadesinde gender=1 ve age, 65'den büyük veya eşit olduğunda doğru yani 1 olur ve seniorFemales'i bir arttırır. Eğer iki durumdan birisi veya her ikisi doğru olmazsa if ifadesinin içi sıfır olur ve seniorFemales bir arttırmadan bir sonraki komuta geçilir.

mantıksal VEYA

mantıksal VEYA (||) işlemi

mantıksal VEYA işleminin doğruluk tablosu şu şekildedir;

ifade1	ifade2	ifade1 || ifade2
0	0	0
0	sıfırdan farklı	1
sıfırdan farklı	0	1
sıfırdan farklı	sıfırdan farklı	1

ifade1 ve ifade2 her ikiside sıfır olursa sonuç sıfır olur diğer durumlarda sonuç birdir.

mantıksal VEYA

if(semesterAverage>=90 || finalExam>=90)

puts(“ogrencinin notu A dir”);

semesterAverage, 90 dan büyük veya eşit veya finalExam, 90 dan büyük veya eşit olduğunda ogrencinin notu A dir yazar.

&& işlemi, || işleminden daha yüksek önceliğe sahiptir.

mantıksal DEĞİL

Tek bir koşula sahiptir, bu nedenle tekil bir işlemdir. Doğru’yu (1), yanlış (0) yapar, yanlışı (0), doğru (1) yapar.

mantıksal DEĞİL

Doğruluk tablosu şu şekildedir;

ifade	! ifade
0	1
sıfırdan farklı	0

işlem önceliği şu şekildedir:

İşlemler	Yön	Tür
++ *(son ek)* -- *(son ek)*	sağdan sola	son ek
+ - ! ++ *(ön ek)* -- *(ön ek)* *(tür)*	sağdan sola	tekil
* / %	soldan sağa	çarpımsal
+ -	soldan sağa	toplamsal
< <= > >=	soldan sağa	ilişkisel
== !=	soldan sağa	eşitlik
&&	soldan sağa	mantıksal VE
||	soldan sağa	mantıksal VEYA
?:	sağdan sola	koşullu
= += -= *= /= %=	sağdan sola	atama
,	soldan sağa	virgül