

Bölüm 4

Program Denetim Deyimleri

İçindekiler

4.1	Algoritma ve Akış Diyagramı	27
4.2	Karşılaştırma Deyimleri	28
4.2.1	if ve if-else Yapısı	28
4.2.2	switch Yapısı	29
4.2.3	İç İçe if Yapıları (Nested if Blocks)	31
4.3	sizeof Fonksiyonu	32
4.4	?: Operatörü	32

4.1 Algoritma ve Akış Diyagramı

Herhangibir programlama dilinde bir program yazabilmek için algoritma bilmek son derece önemlidir. Algoritma, verilen bir problemi çözebilmek için neler yapılması gerektiğini ve işlem adımlarını programlama dilininin yapısı kullanarak göstermektir.

Program yazmadan önce, program yazımını kolaylaştırmak için uygulanması gereken önemli ve ilk adımlardan biridir. Bir programı yazmaya geçmeden önce problemin çözümünün adım adım belirlenmesi gerekiyor. İşlem sırasının belirlenmesinin ardından algoritma C diline daha kolay aktarılabilir.

Akış diyagramı da algoritma ile aynı anlama gelmektedir. Sadece işlem adımlarının gösteriliminde kabaca C dilini kullanmak yerine birtakım anlamalara gelen şekiller kullanılmaktadır.

4.2 Karşılaştırma Deyimleri

Karşılaştırma Deyimleri, bir veya birden fazla ifadenin, belirli koşul sağlandığında yürütülmesi için kullanılır.

Karşılaştırma işlemlerinde koşul, karşılaştırma operatörleri kullanılarak yapılır. Eğer birden fazla karşılaştırma eylemi olacaksa, mantıksal operatörler aracılığı ile karşılaştırma eylemi gerçekleştirilir.

`if` ve `switch` olmak üzere iki tip karşılaştırma deyimi vardır. `?:` operatörü de karşılaştırma işleminde kullanılabilir. Ancak bu bir operatördür, diğerleri gibi fonksiyon değildir. `if` karşılaştırma deyimi, `if-else-if` yapıları şeklinde de kullanılabilir. `switch` deyimi ise `if` deyiminden farklıdır. `switch` deyimi bir değişkenin içeriğine göre programın akışını yönlendirir.

4.2.1 if ve if-else Yapısı

`if` ve `else` tek bir karşılaştırma deyimi olup `else`'in kullanımı isteğe bağlıdır. Eğer koşul sağlanırsa `if` 'den sonraki işlem yürütülür ve `else`'den sonraki işlem atlanır. Olumsuz ise `if` 'den sonraki kume atlanır, eğer varsa `else`'den sonraki işlem gerçekleşir. Aşağıdaki programları inceleyelim.

```
a = 5;
if (a==5)
printf ("a = %d\n",a);
```

Bu program çalıştırıldığında, ekrana

```
a = 5
```

ifadesini yazacaktır. `a=5` satırında `a` değişkeninin değerine 5 sayısı atanmıştır. Buna göre `a`'nın değeri 5'dir. `if` koşulu, `a` değişkeninin değerinin 5'e eşit olması şeklindedir. `a` değişkeninin değeri 5 olduğu için `5=` deyimi sağlanır ve `if` koşulu TRUE veya 1 değeri döndürür. Bu durumda `if` koşulu sağlandığı için `printf` satırı gerçekleşir ve ekrana `a=5` ifadesi yazılır.

2. program

```
a = 7;
if (a==5)
printf ("a = %d\n",a);
```

şeklinde olsun. Bu program 1. program ile aynıdır. Sadece atama işleminde `a` değişkeninin değeri 7 olarak verilmektedir. Bu durumda `7 = 5` olmadığı için `if` fonksiyonu FALSE yani 0 değerini döndürecek, koşul sağlanmadığı için `printf` satırı atlanacaktır.

3. program

```
a = 7;
if (a==5)
printf ("a = %d\n",a);
else
printf ("a = %d\n",a);
```

Bu program da `else` yapısı kullanılmıştır. `else` "değil ise" anlamında kullanılır. `if` koşulu sağlanır ise `if` koşulunun altındaki işlem yürütülecek, sağlanmaz ise `else` deyiminin hemen altındaki işlem yürütülecektir. Bu programda `a` değişkeninin değeri 7 olduğundan ve $7 = 5$ koşulu sağlanmadığından `if` koşulu sağlanmayacak `else` koşulunun hemen alt satırı çalışacaktır.

`else if` yapısını aşağıdaki örnekler ile açıklayalım. `else if` 'in `else` 'den farkı `else`'den hemen sonra koşul verilmesine gerek yoktur. Çünkü `else` 'in anlamı diğer bütün koşulların olmadığı durumdur. Ancak `else if` 'den hemen sonra koşul vermek mümkündür. Aşağıdaki programları inceleyelim.

```
if (a==5)
    printf ("a=%d\n",a);
else if (a!=6)
    printf ("ELSE:a=%d\n",a);
```

Bu programda `a`'nın değeri 5 olursa ekrana `a=5` ifadesi yazılacaktır. `else if` yerine `else` olsaydı, 5 dışında kalan tüm değerler için *ELSE: a=?* bölümü çalışacaktı. Şimdi ise `else if` ile birlikte yine 5 sayısının dışında kalan tüm değerler için *ELSE: a=?* boluğu çalışacak. Yalnız `else if` içerisinde verilen $a \neq 6$ koşulu ile `a` sayısının 6 sayısına da eşit olmama koşulu var. Bu durumda 5 ve 6 sayısının dışında kalan tüm değerler için `else if` satırının altında kalan satır işleyecektir. Eğer `a` sayısı 6 değerine eşit olmuş olsaydı, ekrana hiçbir değer yazılmayacaktı.

4.2.2 switch Yapısı

Bir değişkenin içeriğine bakarak programın akışını birçok seçenekten birine yönlendiren karşılaştırma deyimidir.

```
switch(değişken)
{
    case sabit1:
        .
        .
        .
    case sabit2:
```

```
        .
        .
        .
    default:
        .
        .
        .
}
```

Burada karşılaştırma eylemi *değişken* ve *sabit1*, *sabit2* değişkenleri arasında yapılacaktır. Karşılaştırılmasını istediğimiz değişkeni `switch` 'den hemen sonra parantezler arasında veriyoruz. Durumlar ise case'lerden sonra veriliyor. `switch` deyimi hakkında aşağıdakiler söylenebilir.

- Değişkene atanan sabit değerler ile case'lerden sonraki sabit değerler sırası ile karşılaştırılır. Eşitlik sağlanırsa eşitliği sağlayan case'lere ait deyim yapılır. Eşitlik sağlanmaz ise **default** satırının altındaki satırlar yürütülür.
- `switch` sadece eşitlik karşılaştırması yapabilir ¹ ve case'in karşısına sadece bir değişken yazılabilir. Birden fazla değişken ile karşılaştırma işlemi yapılacak ise o sayıda case kullanılır.
- `switch` deyiminde sadece tamsayı ve karakter tipindeki değişkenlerin karşılaştırılması yapılabilir. Ondalıklı sayılarda karşılaştırılma yapılamaz.
- **break** bir anahtar sözcüğüdür. Dallanmanın gerçekleştiği yerdeki deyimler veya fonksiyonlar yürütüldükten sonra, diğer case'lere ait deyimleri yürütmek içindir.

```
char kr
printf("bir karakter giriniz\n");
kr = getchar();
switch (kr)
{
    case 'a':
    case 'r':
    case 'c':
        printf("Merhaba C\n");
        break:
    case 'b':
```

¹if'lerde eşitlik dışındaki karşılaştırmalar da yapılabilir.

```

case 'c':
    printf("Merhaba D\n");
case 'k':
    printf("Merhaba E\n");
    break:
default:
    printf("a, r, c, b veya k'ya basınız\n");
}

```

Girilen karakter *a*, *r* yada *c*'ye eşit ise yani $kr=a,r$ veya *c* ise *Merhaba C*, *b* veya *c*'ye eşit ise *Merhaba D* ve *Merhaba E* yazılır. *break* deyimi olmadığı için *break* deyimini görene kadar iki *printf* satırında yürütülür. *k*'ya eşit ise *Merhaba E* yazılır. Hiçbiri ise *default* yürütülür ve *a, r, c, b veya k'ya basınız* yazılır.

- *case* işlemini *break* görene kadar sürdürür. *break* gördüğü zaman *case* işlemi sonlanır.

4.2.3 İç İçe if Yapıları (Nested if Blocks)

if yapılarını iç içe kullanmak mümkündür. İç içe *if* kullanmak yerine, duruma göre bu *if* satırlarını tek satırda sadece bir *if* kullanarak ifade etmek de mümkündür. Sözgelimi,

```

if (a==5)
    if(a!=4)
        if(a>=0)
            printf("BURADAYIM!!!\n");

```

bu program da iç içe 3 adet *if* kullanılmıştır. *if* satırları koşul sayısını vermektedir. Tüm bu koşulları tek bir satırda *if* kullanarak ifade etmek mümkündür. BURADAYIM!!! ifadesinin ekrana yazılabilmesi için bu 3 *if* koşulunun da sağlanması gerekmektedir.

```

if ((a==5) && (a!=4) && (a>=0))
    printf("BURADAYIM!!!\n");

```

Bu programda yukarıdaki program tek bir *if* satırı içerisine alınmıştır. Koşulların arası *&&* simgesi yani VE bağlacı ² ile tek bir *if* içinde ifade edilebilir.

²Mantıksal operatörler (Logical Operators); daha ayrıntılı bilgi elde edebilmek için kitaplara bakınız

4.3 sizeof Fonksiyonu

sizeof fonksiyonu kullanılan değişkenlerin bellekte ne kadar yer işgal ettiğini öğrenmek için kullanılır. sizeof bir değişken adını argüman olarak alır ve, o değişkenin işgal ettiği yeri byte olarak verir. Kullanımı,

```
sizeof(değişkenadi)
```

şeklinde dir. C dilinde kullanılan tüm değişken tipleri bellekte yer işgal ederler. Bu yerlerin kaç byte değerine karşılık geldiğini öğrenmek için aşağıdaki programa benzer programlar yazarak öğrenebilirsiniz.

```
printf("sizeof(int) = %d\n",sizeof(int));
```

4.4 ?: Operatörü

Karşılaştırma operatördür. 3 argüman alır. 1. si koşul, diğeri TRUE olması durumunda işletmek istediğimiz deyim 3.su ise FALSE olması durumunda işletmek istediğimiz deyimdir. Buna göre, ilk koşul ?'den hemen önce verilir. ?'den hemen sonra koşul eğer doğrusu ise ne yapılmak istendiği verilir. :'den hemen sonrasında da koşul eğer yanlış olursa ne yapılması istendiği belirtilir. Aşağıdaki programı inceleyelim.

```
printf( "%s\n", not >= 60 ? "Gecti" : "Kaldi" );
```

Bu program çalıştırıldığında not değişkeni ile 60 , sayısı karşılaştırılacaktır. Eğer not değişkeni 60'dan büyük ise Gecti, küçük ise Kaldi bölümü ekrana yazılacaktır. Burada Gecti ifadesi koşulun TRUE olması durumunda çalışacak bölümü, Gecti ifadesi koşulun FALSE olması durumunda çalışacak bölümüdür.

Yukarıdaki program aşağıdaki biçimde de yazılabilir.

```
not >= 60 ? printf("Gecti\n"): printf("Kaldi\n");
```

Programın çalışma mantığı yine yukarıda anlatıldığı gibidir.