BMM 111
Bilgisayar Programlama-I

5. Ders

Dr. Öğr. Üyesi Mustafa İSTANBULLU

Çukurova Üniversitesi
Mühendislik Fakültesi
Biyomedikal Müh. Böl.

E-mail:mm.istanbullu@gmail.com

Not: Slaytlar, kaynakça bölümünde verilen listeden faydalanılarak hazırlanmıştır.

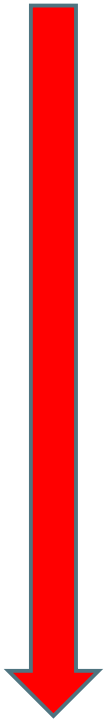
4. Bölüm Seçme Komutları

4.1. Giriş

- Önceki bölümlerde C programlama dilinin ana yapısını, atama komutunu ve girdi-çıktı fonksiyonlarından **`scanf()`** ve **`printf()`**'i gördük.
- Bu öğrendiklerimizle ancak sıralı yapıya sahip bir program yazabiliriz.
- Sıralı akış (sequential flow) özelliğine sahip programlarda komutlar ilk komuttan son komuta kadar, hiçbir komut atlamaksızın sırasıyla yürütülür.

4. Bölüm Seçme Komutları

```
# include <stdio.h>
int main(void)
{  int a=285;
   double y=-27.3789;
   printf("%6.2f\n",y);
   printf("%7.1f\n",y);
   printf("%4d\n",a);
   printf("%2d",a);
   system("PAUSE");
   return(0);
}
```



Komut satırları ATLANMAKSIZIN sırayla yukarıdan aşağıya doğru yürütülür…

Sıralı akış (sequential flow)

4. Bölüm Seçme Komutları

- Ancak, problem çözme sürecinde program akışını değiştirecek farklı komutlara da ihtiyaç duyulur.
- Bu komutların program akışını kontrol etmesi nedeniyle, bunlara kontrol komutları (control statements) adı verilir. Kontrol komutları her programlama dilinde bulunur.
- Kontrol komutları seçme (selection) ve döngü (loop) komutları olmak üzere iki grupta toplanır.
- Bu bölümde C dilinin seçme komutlarını, takip eden bölümde ise döngü komutları ele alınacaktır.
- Seçme komutlarında, öngörülen koşula göre programda hangi komutların yürütüleceği belirlenir. Böylece programcı, program akışını kontrol altına alabilir.
- C dilinde bu seçme işlemini sağlayan **`if`**, **`if-else`** ve **`switch`** komutları bulunmaktadır. Bu komutlara geçmeden önce C dilinde, koşul ifadelerinin nasıl oluşturulduğunu ele alalım.

4.2 Koşul İfadesi

- Koşul ifadeleri ilişkisel (relational) ve mantıksal (logical) operatörler kullanılarak oluşturulur.
- İlişkisel operatörler, değerlerin karşılaştırılmasında kullanılır.
- Örneğin, bir kişinin vücut ısısı `37` dereceden büyük ise, bu kişinin ateşinin olduğunu söyleriz.
- Eğer, kişinin vücut ısısını `int` tipindeki `kisi_derece` değişkeninde saklarsak, bu değişkenin `37` ile karşılaştırılmasıyla aşağıdaki gibi bir koşul ifadesi elde ederiz:

```
kisi_derece > 37
```

- Oluşturulmuş bir koşul ifadesinin sonucu doğru veya yanlıştır.
- C dilinde doğru koşul ifadesi `1` ile değerlendirilirken, yanlış koşul ifadesi `0` ile değerlendirilir.
- Yukarıda verilen örnekte, `kisi_derece` değişkeninin değeri `37`'den büyük ise koşul ifadesinin değeri `1` (doğru); küçük veya eşit ise `0` (yanlış) olacaktır.
- C dilinin ilişkisel operatörlerinden biri olan ve yukarıdaki koşul ifadesinde karşılaştırma işlemini yapan `'>'` sembolü büyüktür anlamına gelmektedir.

- Diğer C ilişkisel operatörleri aşağıdaki tabloda verilmiştir.

İlişkisel Operatör	Anlamı
>	Büyük
>=	Büyük veya eşit
<	Küçük
<=	Küçük veya eşit
==	Eşit
!=	Eşit değil

Örnek

```
int sayac=12;
double alfa=20.5;
Char ch='H';
```

Koşul İfadesi	Sonuç
`(21+sayac) <= 50`	1 (doğru)
`alfa == 65.8`	0 (yanlış)
`sayac >= 12`	1 (doğru)
`ch == 'h'`	0 (yanlış)
`ch != 'h'`	1 (doğru)
`alfa < (sayac-1)`	0 (yanlış)

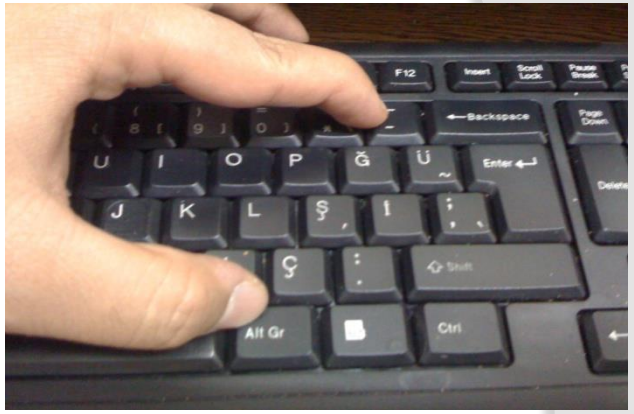
4. Bölüm Seçme Komutları

- Daha önce öğrenilen operatörleri (aritmetik ve atama) de içine alacak şekilde, şu ana kadar bahsedilen C dilindeki tüm operatörlerin öncelik sıraları aşağıdaki tabloda verilmiştir:

Öncelik Sırası	Operatör	Özellik
En Yüksek ↓ En Düşük	( )	İçten dışa
	– +	Tekli operatör (sağdan sola)
	* / %	İkili operatör (soldan sağa)
	+ –	İkili operatör (soldan sağa)
	< <= > >=	İkili operatör (soldan sağa)
	== !=	İkili operatör (soldan sağa)
	=	İkili operatör (sağdan sola)

4. Bölüm Seçme Komutları

- C dilinde ayrıca, mantıksal işlemleri gerçekleştirmek amacıyla
- `'&&'` (mantıksal ve),
- `'||'` (mantıksal veya) ve
- `'!'` (mantıksal değil) mantıksal operatörleri de tanımlanmıştır.



- Bu mantıksal operatörleri ve ilişkisel operatörlerden oluşturulmuş basit koşul ifadelerini biraraya getirerek daha karmaşık koşul ifadeleri oluşturabiliriz.
- Buna göre `'&&'`, `'||'` ve `'!'` operatörlerinin genel kullanımı aşağıdaki gibidir:

```
ifade1 && ifade2
ifade1 || ifade2
! ifade
```

Burada `'&&'` ve `'||'` ikili operatör iken, `'!'` tekli operatördür.

- Az önce bahsedilen operatörlerin doğruluk tabloları aşağıda verilmiştir.
- Aşağıdaki ifadelerde 'sıfır hariç' olarak yapılan tanımlamaya dikkat edelim. Şimdiye kadar doğru için `1`, yanlış için `0` değerini kullandık.
- Gerçekte C dilinde sıfırdan farklı her değer doğru olarak kabul edilmektedir. Bu nedenle doğru ifadesi "sıfırdan hariç" olarak da nitelendirilir.

`&&` Operatörü

ifade 1	*ifade 2*	*ifade 1 && ifade 2*
Sıfır hariç (doğru)	Sıfır hariç (doğru)	1 (doğru)
Sıfır hariç (doğru)	0 (yanlış)	0 (yanlış)
0 (yanlış)	Sıfır hariç (doğru)	0 (yanlış)
0 (yanlış)	0 (yanlış)	0 (yanlış)

- Görüldüğü gibi "***ifade 1 && ifade 2*** " ifadesi, `1` (doğru) değerini ancak her iki ifade de doğru olduğu zaman almaktadır. Diğer durumlarda ise `0` (yanlış) değerini alır.

- || Operatörü

ifade 1	*ifade 2*	*ifade 1* || *ifade 2*
Sıfır hariç (doğru)	Sıfır hariç (doğru)	1 (doğru)
Sıfır hariç (doğru)	0 (yanlış)	1 (doğru)
0 (yanlış)	Sıfır hariç (doğru)	1 (doğru)
0 (yanlış)	0 (yanlış)	0 (yanlış)

- "***ifade 1 || ifade 2***" ifadesi, 0 (yanlış) değerini her iki ifadenin de yanlış olma durumunda alırken, diğer durumlarda 1 (doğru) değerini alır.

- ! Operatörü

ifade	! *ifade*
Sıfır hariç (doğru)	0 (yanlış)
0 (yanlış)	1 (doğru)

- "**! *ifade***" tanımında, yürütülen ifade doğru ise sonuç **0** (yanlış); yanlış ise sonuç **1** (doğru) değerini alır.
- Şimdi, bu operatörlerin kullanımıyla ilgili bazı basit örnekleri inceleyelim.

Örnek

```
int a=8,b=-2, c=0;
double d=7.5; char ch='H';
```

Koşul İfadesi	Sonuç
`(a<5.0)&&(d>b/2)`	0 (yanlış)
`! (b!=8.5)`	0 (yanlış)
`(c>=10) || (ch=='h')`	0 (yanlış)
`!(d-2.5<0)`	1 (doğru)
`(c==0) && (d!=0)`	1 (doğru)
`(ch=='y') || (ch=='Y')`	0 (yanlış)
`a && (a<10)`	1 (doğru)

- Mantıksal operatörleri ile birlikte şu ana kadar öğrenilen tüm operatörlerin öncelik sıralaması aşağıdaki tabloda verilmiştir.

Öncelik Sırası	Operatör	Özellik
En Yüksek ↓ En Düşük	( )	İçten dışa
	– + !	Tekli operatör (sağdan sola)
	* / %	İkili operatör (soldan sağa)
	+ –	İkili operatör (soldan sağa)
	< <= > >=	İkili operatör (soldan sağa)
	== !=	İkili operatör (soldan sağa)
	&&	İkili operatör (soldan sağa)
	||	İkili operatör (soldan sağa)
	=	İkili operatör (sağdan sola)

4.3 `if` Komutu

- Daha önce de belirttiğimiz gibi **`if`** komutları C programlama dilinin seçme komutlarıdır.
- **`if`** komutu, belirtilen koşul altında hangi komut veya komutların yürütüleceğini belirlemeye olanak sağlayan önemli bir komuttur.
- **`if`** komutunun iki ana yapısı bulunmaktadır.
- Bunlar tek yönlü **`if`** ve çift yönlü **`if-else`**' dir.

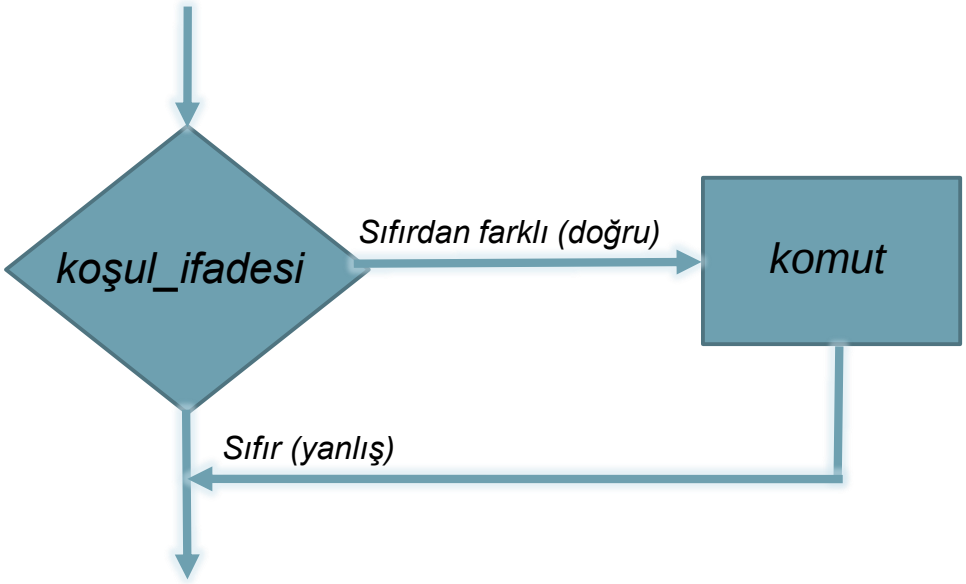
4.3.1 Tek Yönlü `if`

- Tek yönlü `if`'in genel yapısı şu şekildedir:

```
if (koşul_ifadesi)
    komut;
```

- Bu yapıda parantez içine yazılacak ***koşul_ifadesi***, `1` (doğru) veya `0` (yanlış) değerlerinden birini verecek bir koşul ifadesi olmalıdır.
- Bu `if` komutunda ilk önce koşul ifadesinin değeri hesaplanır. Eğer bu koşul ifadesinin değeri sıfırdan farklı (doğru) ise ***komut*** yürütülür.
- Eğer bu koşul ifadesinin değeri `0` (yanlış) ise ***komut*** yürütülmez.
- Her iki durumda da programın kontrolü `if` yapısından sonra yer alan diğer program komutlarına geçer.

- Tek yönlü `if` komutunun program akışı aşağıdaki şemada gösterilmiştir.



Örnek

Bu komutla ilgili aşağıdaki örneği ele alalım…

```
1  printf("Bir sayi giriniz:");
2  scanf("%d",&sayi);
3  if (sayi>0)
4     printf("%d pozitif sayidir.\n",sayi);
5  printf("-----");
```

Bu program parçasında, sayi değeri olarak `15` girildiğini düşünelim. Bu durumda ekran görüntümüz aşağıdaki gibidir.

```
Bir sayi giriniz:15
15 pozitif sayidir.
-----
```

4.3.2 Çift Yönlü `if`

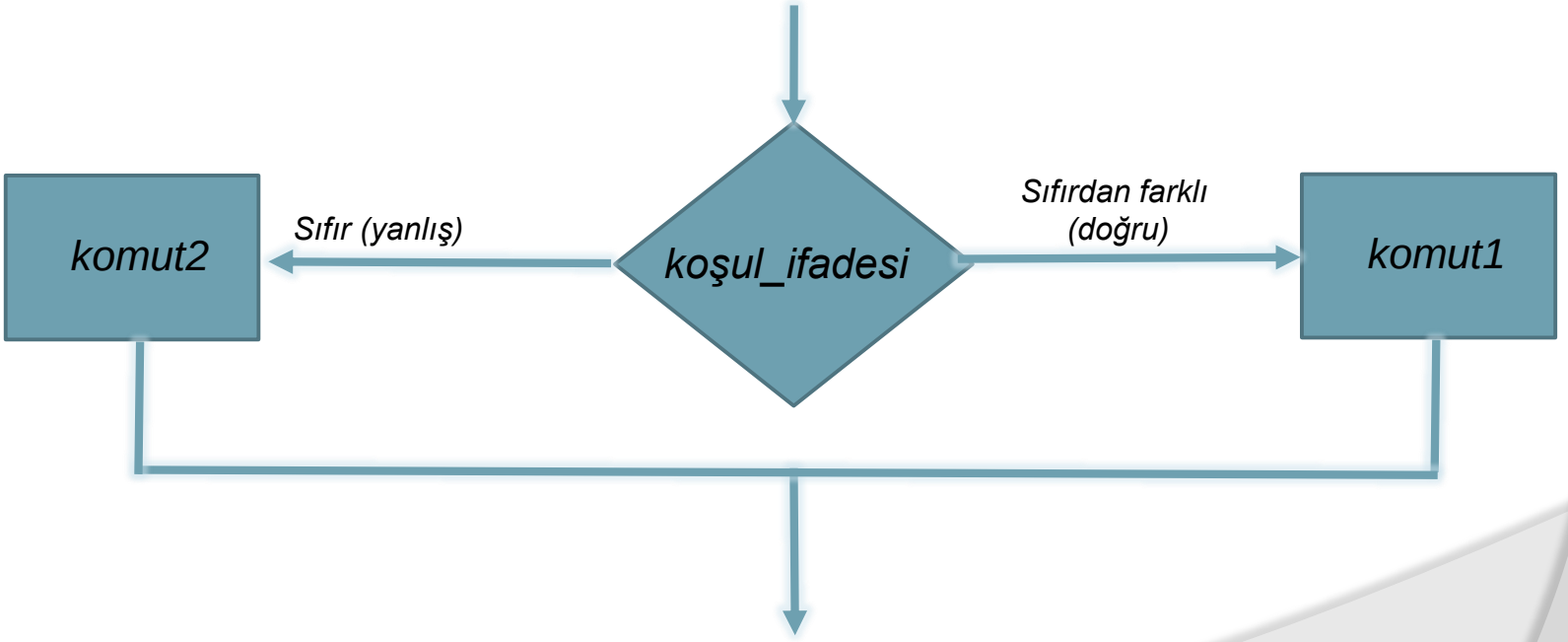
- Çift yönlü `if`'in genel yapısı şu şekildedir:

```
if (koşul_ifadesi)
        komut1;
else
        komut2;
```

- Yine bu yapıda, parantez içine yazılacak ***koşul_ifadesi***'nin değeri sıfırdan farklı (doğru) veya sıfır (yanlış) olabilir. ***komut1*** ve ***komut2*** ise herhangi bir C komutu olabilir. ***komut1*** ve ***komut2*** nin sonlarındaki ';' işareti unutulmamalıdır.

- Bu yapıda *(koşul_ifadesi)* sıfırdan farklı (doğru) ise ***komut1*** yürütülür.; *(koşul_ifadesi)* sıfır ise ***komut2*** yürütülür. Bu komutlardan herhangi biri yürütüldükten sonra programın akışı bir sonraki komuta geçer. Dolayısıyla her iki komutun aynı anda yürütülmesi mümkün değildir.

- Çift yönlü `if` komutunun program akışı aşağıdaki şemada gösterilmiştir.



Örnek

Bu komutla ilgili aşağıdaki örneği ele alalım…

```
1  printf("Bir sayi giriniz:");
2  scanf("%d",&sayi);
3  if (sayi>0)
4      sonuc=sayi*5;
5  else
6      sonuc=2-sayi;
7  printf("Sonuc: %d", sonuc);
```

Bu program parçasında, sayi değeri olarak 3 girildiğini düşünelim. Bu durumda ekran görüntümüz aşağıdaki gibidir.

```
Bir sayi giriniz:3
Sonuc: 15
```

4.4 Bileşik Komut

- C programlama dilinde komutlar, noktalı virgül ';' ile birbirinden ayrılır.
- Bir önceki kısımda gördüğümüz **`if`** komutlarında **`if`** veya **`else`** ifadelerinden sonra sadece bir komut yazılabiliyordu.
- Ancak, bazı problemlerde, bu kısımlara birden fazla komut yazmak istenebilir.
- İşte bu özelliği sağlayan yapı bileşik komuttur. Bileşik komut (compound statement), içinde birçok komut bulunduran tek bir komuttur. Genel yapısı:

```
{
        komut1;
        komut2;
          .
          .
          .

        komutn;
}
```

- Bu yapıda noktalı virgül ile ayrılmış komutlar `'{'` ve `'}'` ayraçları arasında yer almaktadır. Örneğin;

```
{
        printf("Sayi giriniz: ");
        scanf("%d",&a);
        printf("%d sayisinin karesi %d",a,a*a);
}
```

- bir bileşik komuttur.
- Bileşik komutların en çok kullanıldığı yerlerden biri `if` komutlarıdır.
- Bir önceki bölümde verilen `if` ve `if-else` komutlarında yer alan *komut*, *komut1* ve *komut2*'ler birer bileşik komut olabilirler. Örneğin:

- Örneğin;

```
if (x>0)
        printf("%8.3f\n", sqrt(x));
else
        {       y=abs(x);
                printf("%8.3f\n",sqrt(y));
        }
```

- Bu örnekte **x** değişkeninin değeri pozitif ise, **x** değerinin karekökü görüntülenecektir.
- Eğer **x** değişkeninin değeri negatif ise, else kısmında yer alan bileşik komut yürütülecektir.
- Yani, **x** değerinin mutlak değeri ilk önce **y**'ye atanacak ve arkasından **y**'nin karekökü **printf( )** fonksiyonu ile ekranda görüntülenecektir.

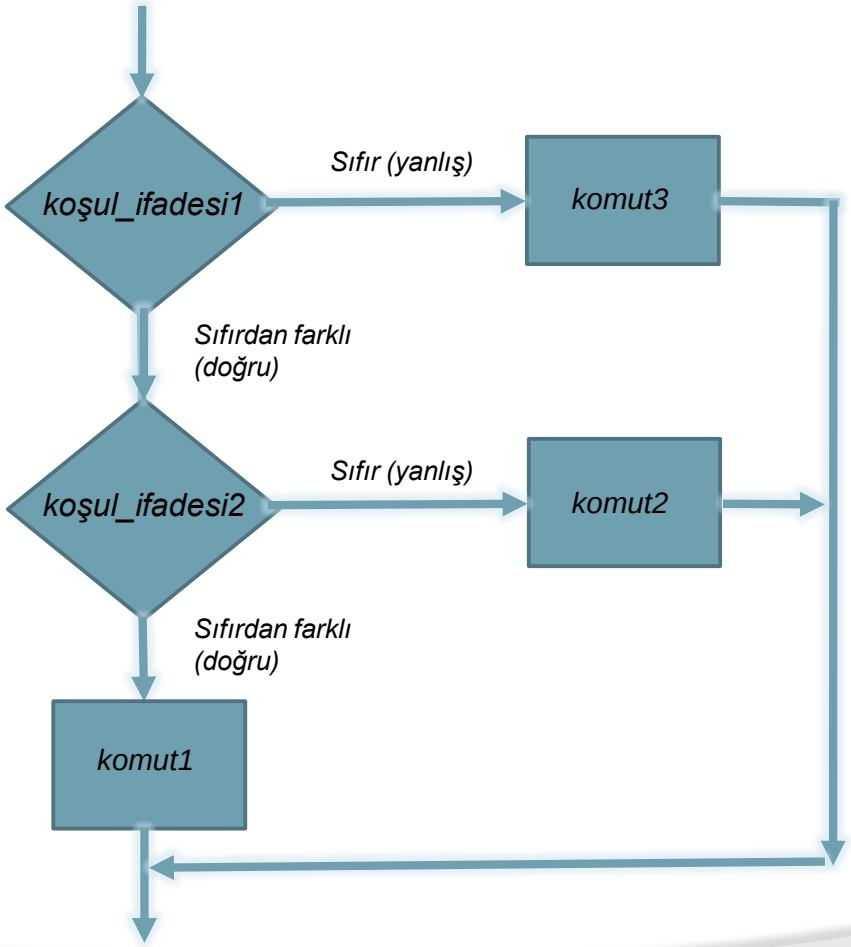
4.5 İçiçe `if` Komutu

- İçiçe if komutlarını (nested if statement) farklı durumlar için inceleyelim:

Örnek Durum 1

```
if (koşul_ifadesi1)
        if (koşul_ifadesi2)
            komut1;
        else
            komut2;
else
        komut3;
```

- İçiçe `if` komutuna ait Örnek Durum 1 için akış şeması şu şekildedir:



Akış şemasından da görüleceği gibi eğer *koşul_ifadesi1* doğru ise, içte yer alan `if-else` komutu, yanlış ise *komut3* yürütülecektir.

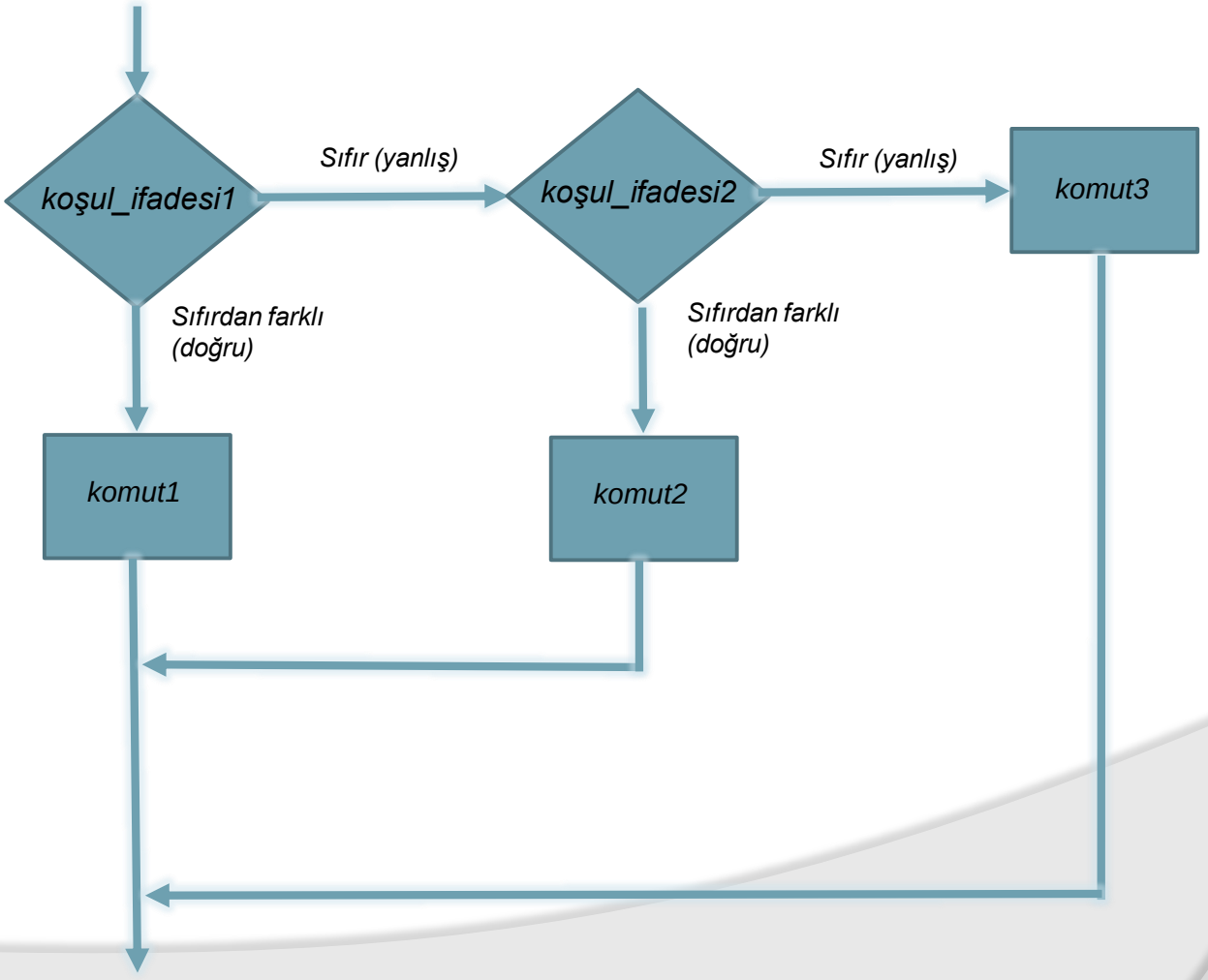
koşul_ifadesi1 ve *koşul_ifadesi2* doğru olduğu durumda *komut1* yürütülürken, *koşul_ifadesi1* doğru, *koşul_ifadesi2* yanlış olduğu durumda *komut2* yürütülecektir.

Örnek Durum 2

- İçiçe if yapısını farklı bir durumda inceleyelim:

```
if (koşul_ifadesi1)
        komut1;
else
        if (koşul_ifadesi2)
                komut2;
        else
                komut3;
```

- İçiçe `if` komutuna ait Örnek Durum 2 için akış şeması şu şekildedir:



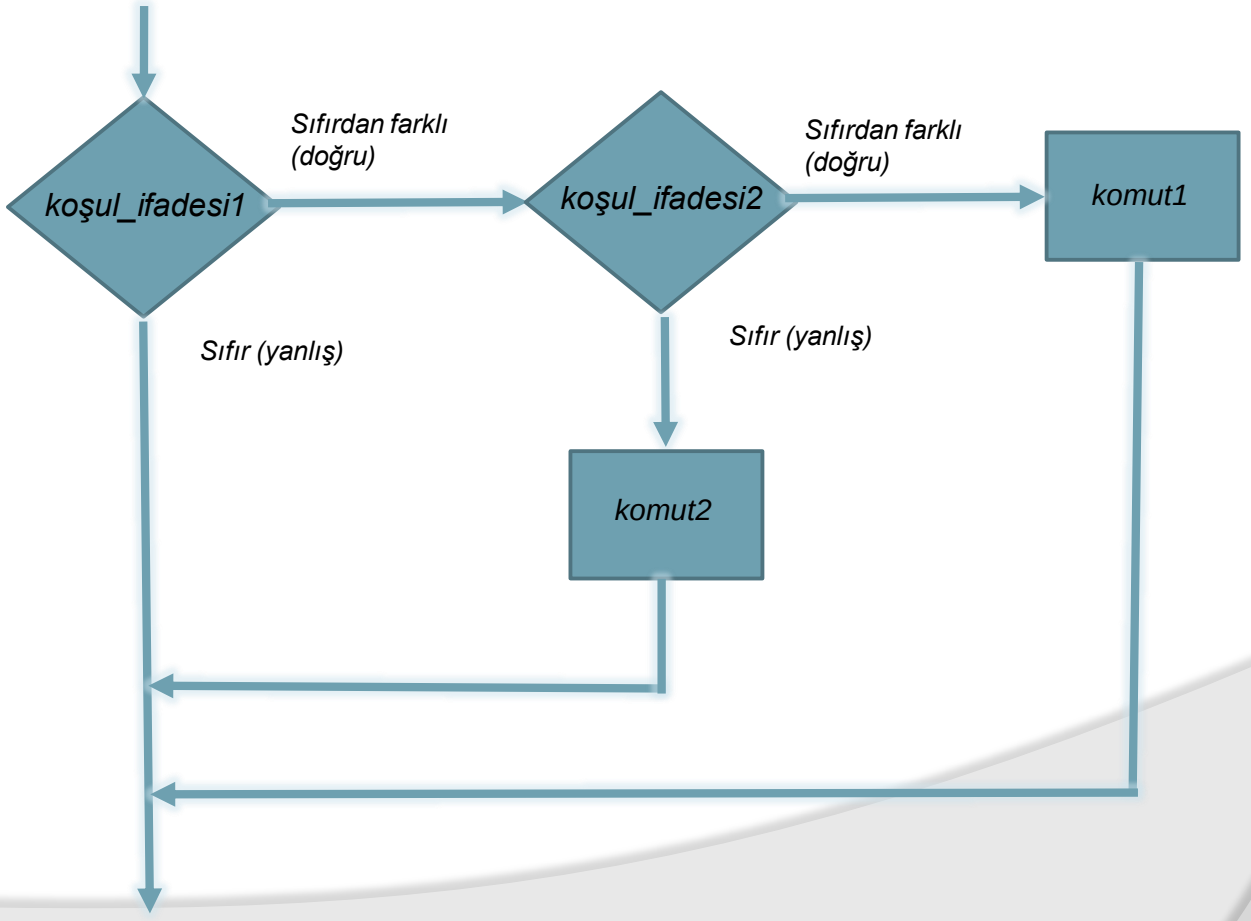
Örnek Durum 3

- Aşağıda bir başka örnek durum verilmektedir.

```
if (koşul_ifadesi1)
        if (koşul_ifadesi2)
                komut1;
        else
                komut2;
```

Bu örnekte, dış kutuda **`else`** kısmı bulunmayan bir **`if`** komutu bulunmaktadır. Bu durumda *koşul_ifadesi1* doğru ise iç kutucukta bulunan **`if-else`** komutu yürütülür, yanlış ise işlem bu yapıyı takip eden komutla devam eder.

İçiçe `if` komutuna ait Örnek Durum 3 için akış şeması şu şekildedir:



- Az önce verilen üç duruma benzer daha birçok örnek oluşturulabilir.
- Konunun daha iyi anlaşılması için içiçe if komutu uygulaması ile ilgili olarak verilen `x,y` değerleri için aşağıdaki fonksiyon hesaplaması ele alınsın:

$$f(x,y) = \begin{cases} xy & x \geq 0,\ y \geq 0 \\ x+y & x \geq 0,\ y < 0 \\ y-x & x < 0,\ y \geq 0 \end{cases}$$

- İlk önce içiçe if kullanmadan $f(x,y)$ fonksiyonunun hesaplanması ile ilgili C dili kodunu yazalım.

```
if (x>=0 && y>=0) fonk=x*y;
if (x>=0 && y< 0) fonk=x+y;
if (x< 0 && y>=0) fonk=y-x;
```

Bu örnekte, sırasıyla bütün `if` komutları yürütülecek olup, her `if` komutunda koşul ifadesi hesaplanacak ve hangi koşul ifadesi doğru ise ona ait fonksiyon hesaplaması yapılacaktır.

- Diğer taraftan içiçe `if` ile yazılmış C kodu aşağıdaki gibidir:

```
if (x>=0)
        if (y>=0)
                fonk=x*y;
        else
                fonk=x+y;

else
        fonk=y-x;
```

4.6 `if-else` Eşleşmesi

- C programlama dilinde programlarımızı, boşluklar kullanarak kodlamamız , derleyicinin derlemesini etkilememektedir.
- Daha önce 2. Bölümde de belirttiğimiz gibi girintili yazım sadece program kodunun okunabilirliği ve anlaşılırlığını arttırmak amacıyla yapılmaktadır.
- Özellikle, girintili yazım içiçe `if` komutlarının daha iyi anlaşılmasını sağlamaktadır.
- Aşağıdaki C kodu hiçbir yazım kuralına bağlı kalmaksızın yazılmıştır.

```
if (no_a>0) printf("%d", no_a); else if(no_b<0)
y=no_a*no_b; else y=no_b+no_a;
```

- Bu yazım şekliyle, hangi komutların hangi koşul altında yürütüleceğini hemen söylemek zordur. Oysaki, aynı kod girintili yazım şekliyle daha anlaşılır olacaktır.

```
if (no_a>0)
        printf("%d", no_a);
else
        if(no_b<0)
                y=no_a*no_b;
        else
                y=no_b+no_a;
```

- C programlama dilinde içiçe `if` komutları ile ilgili önemli bir kural bulunmaktadır. Bu kural

"Her `else` en yakın eşleşmemiş `if` ile eşleştirilir."

- Örneğin,

```
if(a>0)
        if(b>0)
                printf("+");
else
        printf("#");
printf("Son");
```

- Program parçasında `a=-5` ve `b=10` değerini aldığını varsayalım. Bu durumda ekranda sadece 'Son' görüntülenecektir.

- Birçok kişi '#Son' çıktısının elde edilebileceğini düşünebilir. Ancak, yukarıda belirtilen kurala göre, `else` dıştaki `if` ile değil içteki `if` ile eşleştirilir.

- Bu durumda az önceki kodun doğru girintili yazım şekli aşağıdaki gibidir:

```
if(a>0)
        if(b>0)
                printf("+");
        else
                printf("#");
printf("Son");
```

En dıştaki `if`'in `else` kısmı bulunmamaktadır.

4.7 Soru İşareti Operatörü

- C dilinde **`if-else`** komutunun daha etkin olarak kullanılmasını sağlayan operatörler de bulunmaktadır.

- **`'?'`** ve **`':'`** operatör çifti bu amaçla kullanılır. Bu operatör çiftinin kullanım şekli aşağıdaki gibidir:

```
ifade1?ifade2 : ifade3
```

- Buradaki ***`ifade1`***, ***`ifade2`*** ve ***`ifade3`*** birer ifadeyi temsil ederler. Bu durumda öncelikle ***`ifade1`***'in değerine bakılır. Bunun sonucu eğer doğru ise, ***`ifade2`***'de tanımlanan işlemler gerçekleştirilir; yanlış ise ***`ifade3`***'e geçilir.

4.8 `switch` Komutu

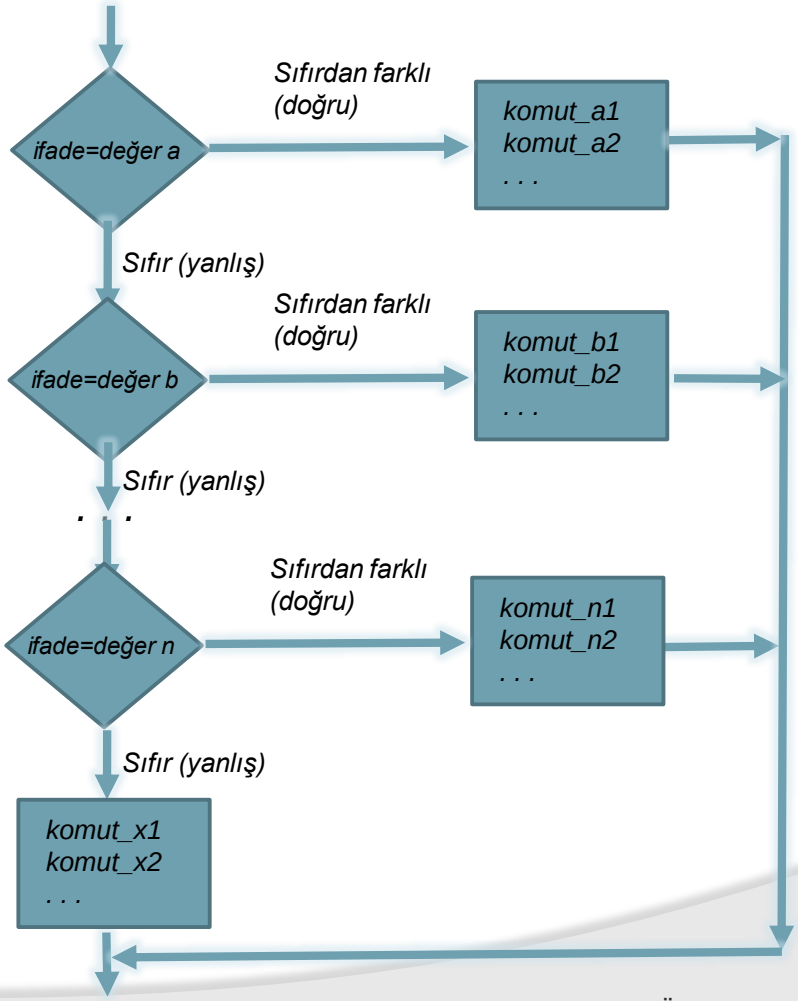
- Önceki kısımlarda, `if` komutunun çoktan seçme problemlerinde nasıl kullanıldığını gördük. `switch` komutu, `if` komutuna alternatif bir seçme komutu olup, genel yapısı aşağıda verilmiştir.

```
switch(ifade)
{ case      değera:  komut_a1;
                     komut_a2;
                        .
                     break;
  case      değerb:  komut_b1;
                     komut_b2;
                        .
                     break;
                        .
  case      değern:  komut_n1;
                     komut_n2;
                        .
                        .
                     break;
   default:          komut_x1;
                     komut_x2;
                        .
                        .
}
```

4. Bölüm Seçme Komutları

- Bu yapıda kullanılan **`switch, case, break ve default,`** dört yeni özel amaçlı sözcüktür.
- **`switch`** sözcüğü, **`switch`** komutunu belirtmektedir. **`switch`** sözcüğünü takip eden parantez içindeki ifade, ilk önce hesaplanır ve ifadenin sonucunda bir tamsayı veya karakter değeri elde edilir.
- Bu ifadenin sonucu daha sonra **`case`** değerleri ile karşılaştırılır.
- Eşit olan değer bulununca takip eden komutlar **`break`** sözcüğüne kadar yürütülür.
- Eğer ifadenin sonucu **`case`**'i takip eden değerlerden birine eşit değilse, **`default`** sözcüğünü takip eden komutlar yürütülür.

- **`switch`** komutunun akış şeması şu şekildedir:



- Az önceki örnekte a değeri 2 girilirse, switch komutunda case değeri 2 olan noktadan başlayarak break komutuna kadar olan kısım yürütülür. Yani, yürütülecek olan satırlar şunlardır:

```
case 2:
case 3:
case 4: printf("0-4 araliginda\n");
        break;
```

- `case 2:` ve `case 3:` satırlarında yürütülecek komut bulunmadığından takip eden `case 4:` teki komut yürütülür ve `break` komutuyla `switch` komutundan çıkılır.

- Sonuçta `"0-4 araliginda"` mesajı görüntülenir. `switch` de kullandığımız `break` aslında bir komut olup "*bulunduğun bloktan çık ve bir sonraki komutla yürütmeye devam et*" anlamını taşımaktadır.

Örnek

`a` değişkeninin değeri `0` olursa program çıktısı ne olur?

```
*8*
```

`a` değişkeninin değeri `1` olursa program çıktısı ne olur?

```
*-7*
```

`a` değişkeninin değeri `2` olursa program çıktısı ne olur?

```
*6*
*8*
```

`a` değişkeninin değeri `3` olursa program çıktısı ne olur?

```
*9*
```

- Şimdiye kadar verilen bütün örneklerde **default** sözcüğü hep **switch** içinde yer aldı.
- Bazen problem yapısına bağlı olarak **default** kısmı kullanılmayabilir.
- Bu durumda **switch**'teki ifade hiçbir **case** değerine eşit olmadığında, **default** kısmı da bulunmayacağından program yürütümü **switch** komutunu takip eden komutla devam edecektir.
- Bu durum aşağıdaki örnekte görülebilir:

```
scanf("%c",&ch);
      switch(ch)
      {     case 'E':
            case 'e':   printf("Erkek\n");
                        break;
            case 'K':
            case 'k':   printf("Kadin\n");
      }
      printf("*********");
```

KAYNAKÇA

- Prof.Dr. İbrahim DEVELİ, Bilgisayar Programlama Ders Notları, Erciyes Üniv. Elektrik-Elektronik Müh. Böl.
- H.Turgut UYAR, Programlamaya Giriş Ders Notları,İTÜ, 2004.
- Fedon KADİFELİ,Standart C Programlama Dili, (Tercüme),2000.
- Doç. Dr. Soner ÇELİKKOL, Programlamaya Giriş ve Algoritmalar, Murathan Yayınevi, TRABZON; 2009
- N. Ercil Çağıltay ve ark., C DERSİ PROGRAMLAMAYA GİRİŞ, Ada Matbaacılık, ANKARA; 2009.
- Çeşitli kişilerin internette paylaşıma açtığı notlardan faydalanılmıştır.