BÖLÜM 6

İLİŞKİSEL VE MANTIKSAL İŞLEÇLER VE SEÇMELİ KONTROL YAPILARI

ANAHTAR KAVRAMLAR

İlişkisel İşleçler, Mantıksal İşleçler, Kontrol Yapıları, If Kontrol Yapısı, If...else Kontrol Yapısı, Switch Kontrol Yapısı, Break Anahtar Kelimesi, Koşullu İşleç

ÖĞRENME HEDEFLERİ

1. İlişkisel işleçleri nasıl kullanacağını bilir
2. Mantıksal işleçleri nasıl kullanacağını bilir
3. İlişkisel ve mantıksal işleçlerin öncelik sıralamasını bilir
4. Program akışının if, if...else gibi yapılarla nasıl kontrol edilebileceğini bilir
5. Aritmetik işleçlerin öncelik sıralamasını bilir

ALTINCI BÖLÜM

GİRİŞ

C++ programlama dilinde ifadeler, farklı bir biçimde yönlendirilmediği sürece ardışık olarak çalıştırılmaktadır. Diğer bir ifadeyle, program önce kod bloğundaki ilk ifadeyi, sonrasında sonlanıncaya kadar sırasıyla diğer ifadeleri çalıştırmaktadır. Bununla birlikte C++ programlama dilinde akış yalnızca ardışık çalıştırma diye adlandırılan bu yöntemle sınırlandırılmamıştır. Çalıştırılması esnasında program, akışı kontrol etmeye imkan veren farklı kontrol yapıları yardımıyla çatallanabilir, belirli kod bloklarını tekrar uygulayabilir ya da farklı seçeneklerden birini çalıştırabilir. C++ programlama dilinin bu esnek yapısı etkin ve kullanışlı kod blokları oluşturulmasına imkan vermektedir. C++ programlama dilinde ardışık, seçmeli ve yinelemeli olmak üzere 3 farklı kontrol yapısı bulunmaktadır. Bu bölümde seçmeli kontrol yapılarından bahsedilecektir. Öncesinde seçmeli kontrol yapılarındaki koşulları oluşturabilmemiz için gerekli olan ilişkisel ve mantıksal işleçlerden bahsedilecektir.

1. İLİŞKİSEL İŞLEÇLER

İlişkisel işleçler, C++ programlama dilinde iki değişken ya da iki değer arasındaki ilişkiyi tanımlayan ya da koşullu ifadelerin bir parçası olarak bu ilişkiyi sınayan yapılardır. İlişkisel işleçler üzerinden oluşturulmuş ilişkisel ifadelerin sonucu "true" (doğru) ya da "false" (yanlış) değeri alan boolean veri tipinde olmaktadır: eğer ifadedeki değerler ya da değişkenler arasında ilişkisel işlecin tanımladığı gibi bir ilişki varsa ifade true boolean değeri almakta, yoksa false boolean değeri almaktadır.

```
int sayi_1 = 2, sayi_2 = 3;

bool buyuktur = (sayi_1 > sayi_2);
```

Örneğin yukarıdaki kod bloğunda sayi_1 ve sayi_2 birer tam sayı değişkeni olarak tanımlanmış ve atama işleci kullanılarak bu değişkenlere sırasıyla 2 ve 3 değerleri atanmıştır. Kod bloğunda 'büyüktür' ilişkisel işleci kullanılarak oluşturulmuş 'sayi_1 > sayi_2' ilişkisel ifadesi yoluyla sayi_1 ve sayi_2 değişkenlerine atanmış değerler kıyaslanmakta ve değerler arasında ifadede belirtildiği gibi bir ilişki olmadığı için, diğer bir ifadeyle sayi_1'e atanmış 2 değeri sayi_2'ye atanmış 3 değerinden büyük olmadığı için false boolean sonuç değeri buyuktur değişkenine atanmaktadır.

Yukarıdaki kullanımının yanı sıra, ilişkisel işleçler aşağıdaki kod bloğunda olduğu gibi koşullu ifadelerin bir parçası olarak da kullanılmaktadır. Kod bloğunda yine yukarıdakine benzer olarak sayi_1 ve sayi_2 birer tam sayı değişkeni olarak tanımlanmış ve atama işleci kullanılarak bu değişkenlerin ilkine 2 değeri ve ikincisine 3 değeri atanmıştır. Burada 'eşit değildir' ilişkisel işleci if koşullu ifadesinin bir parçası olarak sayi_1 ve sayi_2 değişkenlerinin eşit olup olmadığını sınamaktadır. sayi_1 ve sayi_2 değişkenlerine atanmış değerler farklı olduğu için 'sayi_1 ! = sayi_2' ilişkisel ifadesi true değer alacak ve kod çalıştırıldığında ekrana "x ve y esit degildir" ifadesi yazılacaktır.

```
int sayi_1 = 2, sayi_2 = 3;
if (sayi_1 ! = sayi_2){
      cout << "sayi_1 ve sayi_2 esit değildir" <<
endl;
}
```

Aşağıdaki tabloda da belirtildiği gibi; eşittir, eşit değildir, küçüktür, küçük eşittir, büyüktür ve büyük eşittir olmak üzere 6 adet ilişkisel işleç tanımlanmaktadır.

Tablo 6.1 C ++ Programlama Dilinde Tanımlı İlişkisel İşleçler.

İlişkisel İşleç	Sembolü	İlişkisel İfade	İlişkisel İfadenin Anlamı
eşittir	==	x == y	x y'ye eşittir
eşit değildir	!=	x != y	x y'ye eşit değildir
büyüktür	>	x > y	x y'den büyüktür
büyük eşittir	>=	x >= y	x y'den büyük eşittir
küçüktür	<	x < y	x y'den küçüktür
küçük eşittir	<=	x <= y	x y'den küçük eşittir

Diğer birçok programlama dilinde olduğu gibi C++ programlama dilinde de iki değerin birbirine eşit olup olmadığını belirlemek için kullanılan eşittir (==) işleci ile tanımlı değişkenlere değer atanırken kullanılan atama işleci (=) sık sık karıştırılmaktadır. Program tasarlanırken bu durum dikkate alınmalı ve ilgili işleçler doğru yerde kullanılmalıdır.

DİKKAT EDELİM

Eşittir işleci ile atama işleci birbirine karıştırılmamalı ve doğru işlecin doğru yerde kullanımına dikkat edilmelidir.

Örneğin aşağıdaki kod bloğunda sayi_1, sayi_2 ve sayi_3 birer tam sayı değişkeni olarak tanımlanmış ve atama işleci kullanılarak bu değişkelere sırasıyla 3, 4 ve 5 değerleri atanmıştır. İlk koşullu ifadede 'eşittir' ilişkisel işleci sayi_1 ve sayi_3 değişkenlerinin eşit olup olmadığını sorgulamaktadır. sayi_1 ve sayi_3 değişkenlerine atanmış değerler farklı olduğu için 'sayi_1 = = sayi_3' ilişkisel ifadesi false boolean değeri alacak ve ilgili koşullu ifadenin gövdesindeki kod çalıştırılmayacaktır. İkinci koşullu ifadede ise 'küçüktür' ilişkisel işleci ifadedeki ilk değişkenin ikinci değişkenden küçük olup olmadığını sorgulamaktadır. sayi_1 değişkenine atanmış 3 değeri sayi_2 değişkenine atanmış 4 değerinden küçük olduğu için 'sayi_1 < sayi_2' ilişkisel ifadesi true boolean değeri alacak ve ilgili koşullu ifadenin gövdesindeki kod çalıştırılarak ekrana "sayi_1 sayi_2'den kucuktur" cümlesi yazılacaktır. Son olarak üçüncü koşullu ifadede 'büyük eşittir' ilişkisel işleci ifadedeki ilk değişkenin ikinci değişkenden büyük eşit olup olmadığını sorgulamaktadır. sayi_3 değişkenine atanmış 5 değeri sayi_2 değişkenine atanmış 4 değerinden büyük eşit olduğu için 'sayi_3 > = sayi_2' ilişkisel ifadesi true değer alacak ve ilgili koşullu ifadenin gövdesindeki kod çalıştırılarak ekrana "sayi_3 sayi_1'den buyuktur" cümlesi yazılacaktır.

```
int sayi_1 = 3, sayi_2 = 4, sayi_3 = 5;
if (sayi_1 == sayi_3){
      cout << "sayi_1 ve sayi_3 esittir" <<endl;
}
if (sayi_1 < sayi_2){
      cout << "sayi_1 sayi_2'den kucuktur" <<endl;
}
if (sayi_3 >= sayi_2){
      cout << "sayi_3 sayi_1'den buyuktur" <<endl;
}
```

Daha önceki bölümlerde de belirtildiği gibi, C++ programlama dilinde tasarlanan kodun düzenli ve doğru çalışabilmesi için kullanılan işleçler için farklı öncelik sıraları belirlenmiştir. Aritmetik işleçler ilişkisel işleçlere göre daha yüksek önceliğe sahipken, atama işleci ilişkisel işleçlere nispeten daha düşük önceliğe sahiptir. Birden fazla ilişkisel işlecin kullanıldığı ilişkisel ifadelerde işleçlerin kendi içeri-

sinde öncelik sıraları soldan sağa doğru olmaktadır. Doğru ve etkili çalışan bir kod tasarlayabilmek için işleçlerin öncelik sıralarının bilinmesi gerekmektedir. Örneğin aşağıdaki kod bloğunda sayi_1 tamsayı değişkenine atanmış 4 değeri sayi_2 tamsayı değişkenine atanmış 5 değerinden küçük olmasına rağmen, diğer bir ifadeyle 'sayi_1 > sayi_2' ilişkisel ifadesi doğru olmamasına rağmen; aritmetik işleçlerin işlem önceliği olduğu için, '2*sayi_1 > sayi_2 + 2' ilişkisel ifadesinde önce aritmetik işlemler uygulanacak, ve aritmetik işlemlerden sonra 'büyüktür' işlecinin solundaki değer (2*4 = 8) sağındaki değerden (5 + 2 = 7) büyük olacağından '2*sayi_1 > sayi_2 + 2' ilişkisel ifadesi true değer alacaktır. Sonuç olarak ilgili koşullu ifadenin gövdesindeki kod çalıştırılarak ekrana "parantez kullanimi ifadelerin daha anlasilir olmasini saglar" cümlesi yazılacaktır. Her ne kadar aritmetik işleçlerin ilişkisel işleçlerden daha yüksek işlem önceliğine sahip olduğu bilinse de, bunun parantez yoluyla vurgulanması; yani '2*sayi_1 > sayi_2 + 2' ilişkisel ifadesinin parantezler yardımıyla '(2*sayi_1) > (sayi_2 + 2)' şeklinde yazılması, kodun daha anlaşılır ve kolay takip edilmesini sağlayacaktır.

```
int sayi_1 = 4, sayi_2 = 5;
if (2*sayi_1 > sayi_2 + 2){
    cout << "parantez kullanimi ifadelerin daha anlasilir olmasini saglar" << endl;
}
```

DİKKAT EDELİM

Doğru şekilde çalışan kod blokları oluşturabilmek için işleçlerin öncelik sıraları bilinmelidir. Bu noktada parantez kullanımı hangi işleçlerin önce uygulanması gerektiğini vurgulayacak, daha anlaşılır ve takip edilmesi kolay kod blokları oluşturabilmemize imkan sağlayacaktır.

2. MANTIKSAL İŞLEÇLER

İlişkisel işleçler 'not > = 90' veya 'sicaklik < 21' gibi tek bir durumun sınandığı koşullu ifadeler oluşturmak için yeterlidir. Ancak "sıcaklık 30'dan küçük ve nem %80'den fazlaysa" ya da "vize notu 90 ve final notu 80 ise" gibi birden fazla durumun sınandığı daha karmaşık koşullu ifadelerin oluşturulabilmesi mantıksal işleçlerin kullanımıyla mümkün olmaktadır. Aşağıdaki tabloda da belirtildiği gibi C++ programlama dilinde DEĞİL, VE ve VEYA olmak üzere 3 adet mantıksal işleç tanımlanmaktadır.

Tablo 6.2 C ++ Programlama Dilinde Tanımlı Mantıksal İşleçler.

İşleç	Sembolü	Mantıksal İfade	Mantıksal İfadenin Anlamı
DEĞİL	!	!(x < y)	x y'den küçük değildir
VE	&&	(x == y) && (y < z)	x y'ye eşit ve y z'den küçüktür
VEYA	[]	(x > y) [] (y >= z)	x y'den büyük veya y z'den büyük eşittir

```
int sayi_1 = 12, sayi_2;
if (!(sayi_1 < 10)){
        sayi_2 = 10;
}
```

Mantıksal değil işleci ile oluşturulmuş ifadenin boolean değeri, ifadedeki durum doğru olduğunda false, yanlış olduğunda true olacaktır. Örneğin yukarıdaki kod bloğunda sayi_1 ve sayi_2 birer tam sayı değişkeni olarak tanımlanmış ve atama işleci kullanılarak bu değişkenlerden ilkine 12 değeri atanmıştır. Kod bloğunda 'küçüktür' ilişkisel işleci kullanılarak oluşturulmuş 'sayi_1 < 10' ilişkisel ifadesi sayi_1'in değeri 10'dan küçük olmadığı için false boolean değeri alacaktır. 'sayi_1 < 10' ilişkisel ifadesi false boolean değeri aldığı için, mantıksal DEĞİL işleci ile oluşturulmuş '!(sayi_1 < 10)' ifadesi true boolean değeri alacaktır ve sonuç olarak koşullu ifadenin gövdesindeki kod çalıştırılarak sayi_2 değişkenine 10 değeri atanacaktır. Aşağıdaki tabloda herhangi bir durumun alacağı farklı boolean değerlerine karşılık mantıksal DEĞİL işlecinin alacağı boolean değerleri verilmiştir.

Tablo 6.3 Mantıksal DEĞİL İşlecinin Boolean Değerleri Tablosu.

Durum	! Durum
true	false
false	true

Mantıksal VE işleci ile oluşturulmuş ifadenin boolean değeri, ifadedeki durumlardan her ikisi de doğru olduğunda true, diğer bütün durumlarda false olmaktadır. Örneğin aşağıdaki kod bloğunda sayi_1, sayi_2 ve sayi_3 birer tam sayı değişkeni olarak tanımlanmış ve atama işleci kullanılarak bu değişkenlere sırasıyla 1, 2 ve 3 değerleri atanmıştır. Değişkenlerin aldığı 1, 2 ve 3 değerleri göz önünde bulundurulduğunda, kod bloğundaki ilk koşullu ifadede sırasıyla 'büyüktür' ve 'büyük eşittir' ilişkisel işleçleri kullanılarak oluşturulmuş 'sayi_3 > sayi_2' ve 'sayi_2 > = sayi_1' ilişkisel ifadelerinin her ikisi de true boolean değeri alacaktır. Mantıksal VE işlecinin yukarıda verdiğimiz tanımı düşünüldüğünde, ifadedeki durumların her ikisi de true boolean değeri aldığından '(sayi_3 > sayi_2) && (sayi_2 > = sayi_1)' ifadesinin kendisi de true değer alacaktır. Ve sonuç olarak koşullu ifadenin gövdesindeki kod çalıştırılarak ekrana "sayi_3 sayi_1'den buyuktur" cümlesi yazılacaktır.

```
int sayi_1 = 1, sayi_2 = 2, sayi_3 = 3;
if ((sayi_3 > sayi_2) && (sayi_2 > = sayi_1)){
        cout << "sayi_3 sayi_1'den buyuktur" <<endl;
}
if ((sayi_3 > sayi_2) && (sayi_3 < sayi_1)){
        cout << "sayi_1 sayi_3'den buyuktur" <<endl;
}
```

Diğer yandan yukarıdaki kod bloğundaki ikinci koşullu ifadede; 'büyüktür' ilişkisel işleci kullanılarak oluşturulmuş 'sayi_3 > sayi_2' ifadesi true ama 'küçüktür' ilişkisel işleci kullanılarak oluşturulmuş 'sayi_3 < sayi_1' ifadesi false değer alacağından, '(sayi_3 > sayi_2) && (sayi_3 < sayi_1)' mantıksal ifadesi de false boolean değeri alacak ve dolayısıyla ilgili koşullu ifadenin gövdesindeki kod çalıştırılmayacaktır. Aşağıdaki tabloda herhangi iki durumun alacağı farklı bool değerlerine karşılık mantıksal ve işlecinin alacağı boolean değerleri verilmiştir.

Tablo 6.3 Mantıksal VE İşlecinin Boolean Değerleri Tablosu.

Durum_1	Durum_2	Durum_1 || Durum_2
true	true	true
true	false	true
false	true	true
false	false	false

Mantıksal VEYA işleci ile oluşturulmuş ifadenin boolean değeri, ifadedeki durumlardan her ikisi de yanlış olduğunda false, diğer bütün durumlarda true olmaktadır. Örneğin aşağıdaki kod bloğunda sayi_1, sayi_2 ve sayi_3 birer tam sayı değişkeni olarak tanımlanmış ve atama işleci kullanılarak bu değişkenlere sırasıyla 1, 2 ve 3 değerleri atanmıştır. Değişkenlerin aldığı 1, 2 ve 3 değerleri göz önünde bulundurulduğunda, kod bloğundaki ilk koşullu ifadedeki sırasıyla 'eşittir' ve 'büyüktür' ilişkisel işleçleri kullanılarak oluşturulmuş 'sayi_3 = = sayi_2' ve 'sayi_2 > sayi_1' ilişkisel ifadelerinin ilki false ve ikincisi true boolean değeri alacaktır. Mantıksal VEYA işlecinin yukarıda verdiğimiz tanımı düşünüldüğünde, ifadedeki durumların biri true değer aldığından '(sayi_3 = = sayi_2) (sayi_2 > sayi_1)' ifadesi de true değer alacaktır. Ve sonuç olarak koşullu ifadenin gövdesindeki kod çalıştırılarak ekrana "sayi_3 sayi_1'den buyuktur" cümlesi yazılacaktır.

```
int sayi_1 = 1, sayi_2 = 2, sayi_3 = 3;
if ((sayi_3 = = sayi_2) || (sayi_2 > sayi_1)){
      cout << "sayi_3 sayi_1'den buyuktur" <<endl;
}
if ((sayi_3 < sayi_2) || (sayi_3 = = sayi_1)){
      cout << "sayi_1 sayi_3'den buyuktur" <<endl;
}
```

Diğer yandan yukarıdaki kod bloğundaki ikinci koşullu ifadede; 'küçüktür' ilişkisel işleci kullanılarak oluşturulmuş 'sayi_3 < sayi_2' ifadesi ve 'eşittir' ilişkisel işleci kullanılarak oluşturulmuş 'sayi_3 = = sayi_1' ifadesi false değer alacağından, '(sayi_3 < sayi_2) || (sayi_3 = = sayi_1)' mantıksal ifadesi de

false boolean değeri alacak ve dolayısıyla ilgili koşullu ifadenin gövdesindeki kod çalıştırılmayacaktır. Aşağıdaki tabloda herhangi iki durumun alacağı farklı bool değerlerine karşılık mantıksal veya işlecinin alacağı boolean değerleri verilmiştir.

Tablo 6.4 Mantıksal VEYA İşlecinin Boolean Değerleri Tablosu.

Durum_1	Durum_2	Durum_1 && Durum_2
true	true	true
true	false	false
false	true	false
false	false	false

İlişkisel işleçler mantıksal işleçlere göre daha yüksek önceliğe sahipken, atama işleci mantıksal işleçlere nispeten daha düşük önceliğe sahiptir. Birden fazla mantıksal işlecin kullanıldığı ilişkisel ifadelerde işleçlerin kendi içerisinde öncelik sıraları soldan sağa doğru olmaktadır. Örneğin aşağıdaki kod bloğunda koşullu ifadede ilişkisel işleçlerin mantıksal işleçlere nispeten işlem önceliği olduğu için, program önce iki ilişkisel ifadenin boolean değerini hesaplayacak ve sonrasında elde edilen boolean değerlerine mantıksal VE işlecini uygulayacaktır. İlk ilişkisel ifadede; aritmetik işleçlerin işlem önceliği olduğu için önce + işlemi uygulanacak, ve sonrasında 'eşittir' işlecinin solundaki değer (2 + 1 = 3) sağındaki değere (sayi_2 = 3) eşit olduğu için 'sayi_1 + 1 = = sayi_2' ilişkisel ifadesi true değer alacaktır. İkinci ilişkisel ifadede de benzer şekilde; 'küçüktür' işlecinin solundaki değer (sayi_2 = 3) sağındaki değerden (sayi_3 = 5) küçük olduğu için 'sayi_2 < sayi_3' ilişkisel ifadesi true boolean değer alacaktır. Koşullu ifadedeki her iki ilişkisel ifade de true değer aldığından mantıksal VE işlecinin sonucu da true boolean değeri alacak ve ilgili koşullu ifadenin gövdesindeki kod çalıştırılarak ekrana "aritmetik islecler mantiksal isleclere nispeten daha yuksek oncelige sahiptir" cümlesi yazılacaktır.

```
int sayi_1 = 2, sayi_2 = 3, sayi_3 = 5;
if ((sayi_1 + 1 = = sayi_2) && (sayi_2 < sayi_3)){
    cout << "aritmetik islecler mantiksal isleclere nispeten daha yuksek oncelige sahiptir" << endl;
}
```

DİKKAT EDELİM

C++ programlama dilinde mantıksal ifadelerde kısa devre değerlendirme yöntemi uygulandığı unutulmamalı, ifadenin ikinci kısmına kod için kritik sayılabilecek işlemler konulmamalıdır.

Mantıksal VE işlecinin doğruluk değer tablosu incelendiğinde durumlardan ilki false olduğunda ikinci durum ne olursa olsun mantıksal ifadenin boolean değerinin false olduğu gözlemlenecektir. Yine benzer şekilde mantıksal VEYA işlecinin doğruluk değer tablosu incelendiğinde durumlardan ilki true olduğunda ikinci durum ne olursa olsun mantıksal ifadenin boolean değerinin true olduğu

gözlemlenecektir. Bu gerçekten hareketle C++ programlama dilinde mantıksal VE ve VEYA işleçleriyle oluşturulmuş mantıksal ifadelerde ilk durum ifadenin doğruluk değeri için yeterli ise program kısa devre değerlendirme adı verilen bir yöntemle ikinci durumu kontrol etmeden direk sonucu dönecektir. Diğer bir ifadeyle, mantıksal VE işleci ile oluşturulmuş ifadelerde, ilk durum false olduğunda program ikinci kısmı kontrol etmeksizin false boolean değeri; benzer şekilde mantıksal VEYA işleci ile oluşturulmuş ifadelerde, ilk durum true olduğunda program ikinci kısmı kontrol etmeksizin true boolean değeri dönecektir.

DİKKAT EDELİM

Mantıksal işleçlerin yukarıda verilen tanımları ışığında
!(true && false) || (false && true) ifadesinin alacağı boolean değerini belirleyiniz.

3. SEÇMELİ KONTROL YAPILARI

Seçmeli kontrol yapıları programın akışını belli koşullar üzerinden çatallandırmaya imkan veren yapılardır. C++ programlama dilinde if (tek seçim), if...else (ikili seçim) ve switch (çoklu seçim) olmak üzere üç farklı seçmeli kontrol yapısı tanımlanmıştır.

4. TEK SEÇİM KONTROL YAPISI

If kontrol yapısı aşağıdaki formda kullanılmaktadır. Eğer koşul kısmındaki ifade doğru ise gövdedeki kod çalıştırılmakta, aksi durumda gövdedeki kod çalıştırılmamaktadır.

```
if (koşul) {
   kod satırı_1
   kod satırı_2
...
}
```

Aşağıda da gösterildiği gibi tek bir kod satırından oluşuyorsa gövde kısmı parantez kullanılmaksızın da oluşturulabilmektedir.

```
if (koşul)
   kod satırı
```

Örneğin aşağıdaki kod bloğunda sayi_1 tamsayı değişkeni pozitif bir değer almışsa, if kontrol yapısının koşul kısmındaki 'sayi_1 > 0' ilişkisel ifadesi true değer alacak ve gövde kısmındaki kod çalıştırılarak ekrana "sayi_1 pozitif bir tamsayidir" yazdırılacaktır. Aksi durumda, yani sayi_1 tamsayı değişkeni pozitif bir değer almamışsa, 'sayi_1 > 0' ilişkisel ifadesi false değer alacağından koşul sağlanmayacak ve gövde kısmındaki kod çalıştırılmayacaktır.

```
int sayi_1;
if (sayi_1 > 0){
   cout << "sayi_1 pozitif bir tamsayidir" <<endl;
}
```

5. İKİLİ SEÇİM KONTROL YAPISI

If kontrol yapısında gövdedeki kod ancak koşul doğru olduğunda çalıştırılır. Koşul yanlış olduğunda ise program yapının gövdesindeki kodu pas geçerek sonraki kod bloğuyla yoluna devam eder. If yapısından farklı olarak if...else kontrol yapısı ise koşul yanlış olduğunda da belirli bir kod bloğunun çalıştırılabilmesine imkan vermektedir. If...else kontrol yapısı aşağıdaki formda kullanılmaktadır: koşul kısmındaki ifade doğru ise kod_parçası_1 çalıştırılmakta, doğru değilse kod_parçası_2 çalıştırılmaktadır.

```
if (koşul) {
   kod_parçası_1
}
else {
   kod_parçası_2
}
```

Örneğin aşağıdaki kod bloğunda eğer basari_notu tamsayı değişkeninin aldığı değer 60'a eşit veya daha büyükse if...else kontrol yapısının koşul kısmındaki 'basari_notu > = 60' ilişkisel ifadesi true boolean değeri alacak ve gövde kısmındaki kod çalıştırılarak ekrana "dersi gectin" yazdırılacaktır. Aksi durumda, yani başarı_notu tamsayı değişkeni 60'tan daha küçük değer almışsa, 'başarı_notu > = 60' ilişkisel ifadesi false boolean değeri alacağından koşul sağlanmayacak ve alternatif gövdedeki kod çalıştırılarak ekrana "derste basarisiz oldun" ve "bir dahaki sefere daha cok calismalisin" cümleleri yazdırılacaktır.

```
int basari_notu;
if (basari_notu > = 60){
   cout << "dersi gectin" << endl;
}
else {
   cout << "derste basarisiz oldun" << endl;
   cout << "bir dahaki sefere daha cok calismalisin" << endl;
}
```

C++ programlama dilinde koşullu işleç adı verilen ve if...else kontrol yapısı ile aynı işlevi gören daha basit bir işleç de tanımlanmıştır. Koşullu işleç genel haliyle aşağıdaki formda kullanılmaktadır:

```
(koşullu ifade ? kod_1 : kod_2)
```

koşullu ifade doğru ise kod_1, yanlış ise kod_2 çalıştırılmaktadır. If...else yapısı ile oluşturulan yukarıdaki kod bloğunun koşullu işleç ile basit şekilde yazımı aşağıdaki gibi olmaktadır:

```
cout << (basari_notu > = 60 ? "dersi gectin" : "derste basarisiz oldun");
```

6. ÇOKLU SEÇİM KONTROL YAPISI

Yukarıda da ifade edildiği gibi if...else kontrol yapısı tek bir koşulu kontrol etmektedir. Eğer koşul doğruysa program yapının gövdesindeki kodu, yanlışsa alternatif kodu çalıştırmaktadır. Bazı durumlarda birden fazla koşulun test edilmesi ve duruma göre alternatif kod satırlarından birinin seçilmesi gerekebilir. If...else yapısının daha kapsamlısı olarak değerlendirilebilecek if...else if kontrol yapısı birden çok

koşulun test edilebilmesine imkan vermektedir. Yukarıdaki örnekte if...else yapısı kullanılarak öğrencinin dönem sonu başarı notuna göre dersi geçip geçmediğini belirleyen bir kod bloğu incelenmişti. Şimdi dönem sonu başarı notuna göre öğrencinin dersten aldığı harf notunu belirleyen bir program tasarlamaya çalışalım. Harf notu belirlenirken birden fazla koşulun test edilmesi gerektiği için if...else if yapısını kullanmak yerinde olacaktır.

```
int basari_notu;


if (basari_notu > = 90){
        cout << “basarı notuna karsilik gelen harf notu A’dir” << endl;
}
else if(basari_notu > = 80){
        cout << “basarı notuna karsilik gelen harf notu B’dir” << endl;
}
else if(basari_notu > = 70){
        cout << “basarı notuna karsilik gelen harf notu C’dir” << endl;
}
else if(basari_notu > = 60){
        cout << “basarı notuna karsilik gelen harf notu D’dir” << endl;
}
else{
        cout << “basarı notuna karsilik gelen harf notu F’dir” << endl;
}
```

Örneğin öğrencinin dönem sonu başarı notu 92 ise, yukarıdaki kod bloğundaki ilk koşul sağlandığından, yani ‘basari_notu > = 90’ ilişkisel ifadesi true boolean değeri alacağından, koşulun gövdesindeki kod çalıştırılacak ve ekrana “basari notuna karsilik gelen harf notu A’dir” ifadesi yazdırılacaktır. Diğer yandan eğer öğrencinin dönem sonu başarı notu 75 ise, yukarıdaki kod bloğundaki ilk koşul sağlanmayacak ve program ilk koşulun gövdesindeki kodu pas geçerek ikinci koşulu test edecektir. Verilen başarı notu için ikinci koşul da sağlanmadığından, diğer bir ifadeyle ‘basari_notu > = 80’ ilişkisel ifadesi false boolean değeri alacağından, program bir önceki koşula benzer biçimde bu koşulun gövdesindeki kodu da pas geçecek ve bir sonraki koşulu test edecektir. Verilen başarı notu için üçüncü koşul sağlanacağından, yani ‘basari_notu > = 70’ ilişkisel ifadesi true boolean değeri alacağından program bu koşulun gövdesindeki kodu çalıştıracak ve ekrana “basari notuna karsilik gelen harf notu C’dir” cümlesi yazdıracaktır.

Şimdi de taş-kağıt-makas oyunu tasarlamak için kullanılabilecek bir kod bloğu üzerinden if...else if kavramını pekiştirmeye çalışalım. Bilindiği gibi taş-kağıt-makas oyununda oyuncular aynı anda ve rastgele bir biçimde taş, kağıt ve makas alternatiflerinden birini seçer ve seçtiği alternatifi ilgili el işaretiyle karşı oyuncuya bildirirler. Oyunda önceden belirlenmiş kurallara göre (taş makası kırar, kağıt taşı kaplar ve makas kağıdı keser) oyunculardan biri oyunu kazanmış sayılır. Aşağıdaki kod bloğu, birinci oyuncunun makası seçtiği bilindiğinde ikinci oyuncunun seçimine göre kazananı belirlemektedir. Kod bloğunda oyuncu_1 ve oyuncu_2 birer string değişkeni olarak tanımlanmış ve atama işleci kullanılarak oyuncu_1 değişkenine “makas” kelimesi atanmıştır. Eğer oyuncu_2 string değişkeni “tas” değerini al-

mışsa ilk koşul sağlanmış olacak ve koşulun gövdesindeki kod çalıştırılarak ekrana "oyunu ikinci oyuncu kazandi" cümlesi yazdırılacaktır. Eğer oyuncu_2 değişkeninin alacağı değer "makas" ise, if...else if yapısındaki ne ilk ne de ikinci koşul sağlanacağından program ilk iki koşulun gövdesindeki kodu pas geçecek ve alternatif gövdedeki kodu çalıştırarak ekrana "oyuncular berabere kaldilar" cümlesini yazdıracaktır.

```
string oyuncu_1, oyuncu_2;
oyuncu_1 = "makas";
if (oyuncu_2 = = "tas"){
        cout << "oyunu ikinci oyuncu kazandi" << endl;
}
else if(oyuncu_2 = = "kagit"){
        cout << "oyunu birinci oyuncu kazandi" << endl;
}
else{
        cout << "oyuncular berabere kaldilar" << endl;
}
```

C++ programlama dilinde if...else if gibi çoklu seçime imkan bir diğer yapı da switch kontrol yapısıdır. Switch, bir değişkenin ya da ifadenin alabileceği farklı değerlere göre alternatif durumlardan birini seçmemize, diğer bir ifadeyle farklı kod satırlarını çalıştırabilmemize imkan veren seçmeli bir kontrol yapısıdır. Switch kontrol yapısı aşağıda belirtildiği gibi case, break ve default anahtar kelimelerinden oluşmaktadır. Bu kontrol yapısında, kontrol olarak adlandırılan değişken ya da ifade deger_1, deger_2 gibi değerlerden biri ile eşleşiyorsa, ilgili kod parçası break anahtar kelimesine kadar çalıştırılacak; eğer kontrol değişkeni hiç bir değerle eşleşmemişse, default anahtar kelimesi ile belirlenmiş kod parçası çalıştırılacaktır. Break ve default anahtar kelimeleri zorunlu olmamakla birlikte, daha kullanışlı ve etkili switch yapısı oluşturabilmek için genellikle tercih edilmektedirler. Break kontrol değişkeninin kalan değerlerle eşleşip eşleşmediğini test etmeden, yani daha fazla kod çalıştırmadan switch yapısını sonlandırmamıza imkan sağlarken; default kontrol değişkeninin tanımlanan değerlerden biriyle eşleşmediği durumlar için bile uygulanabilecek alternatif kod parçası oluşturmamıza imkan vermektedir.

```
switch(kontrol){
        case deger_1: {
                kod_parcasi_1
                break;
        }
        case deger_2: {
                kod_parcasi_2
                break;
        }
        ...
        default: {
                kod_parcasi*
                break;
        }
}
```

Switch yapısı kullanılarak geliştirilen aşağıdaki kod bloğu, yukarıdakine benzer olarak birinci oyuncunun makası seçtiği bilindiğinde ikinci oyuncunun seçimine göre kazananı belirlemektedir. Yalnız switch kontrol yapısı sadece tamsayı ve karakter veri türleri ile çalıştığı için, oyuncular seçimleri birer string değişkeni olarak değil de char değişkeni olarak tanımlanacaktır. Yani "tas", "kagit" ve "makas" kelimeleri yerine sırasıyla 'T', 'K' ve 'M' karakterleri kullanılacaktır. Örneğin ikinci oyuncu kağıdı seçtiğinde, yani oyuncu_2 değişkeni 'K' değeri aldığında; değişkenin aldığı değer ikinci durumla eşleştiği için, program ikinci gövdedeki kod parçasını break anahtar kelimesine kadar çalıştıracak ve ekrana "oyunu birinci oyuncu kazandi" cümlesini yazdıracaktır. Kod bloğunda break anahtar kelimesi kullanılmasaydı, program oyuncu_2 değişkeninin aldığı değer ikinci durumla eşleştiği için, ikinci gövdeden itibaren yapının sonuna kadar bütün kodları çalıştıracak ve ekrana sırasıyla "oyunu birinci oyuncu kazandi", "oyunucular berabere kaldilar" ve "gecerli deger girilmedi" cümleleri yazdırılacaktı.

```
char oyuncu_1, oyuncu_2;
oyuncu_1 = 'M';
switch(oyuncu_2){
    case 'T': {
        cout << "oyunu ikinci oyuncu kazandi" << endl;
        break;
    }
    case 'K': {
        cout << "oyunu birinci oyuncu kazandi" << endl;
        break;
    }
    case 'M': {
        cout << "oyunucular berabere kaldilar" << endl;
        break;
    }
    default: {
        cout << "gecerli deger girilmedi" << endl;
        break;
    }
}
```

ARAŞTIRALIM

If...else if yapısı kullanılarak oluşturulan ve dönem sonu başarı notuna göre öğrencinin dersten aldığı harf notunu belirleyen kod bloğunu switch kontrol yapısını kullanarak yeniden tasarlayın.

7. İÇ İÇE IF...ELSE IF ÇOKLU SEÇİM KONTROL YAPISI

C++ dilinde bazı durumlarda bir biri içerisinde if...else if kontrol yapıları kullanarak çoklu durumları kontrol etmek ve alternatif kodlardan birini seçmek gerekebilir. Şimdi girilen 2 boolean değerine mantıksal VEYA işlecini uygulayan ve sonucu ekrana yazdıran kod bloğu üzerinden iç içe if...else if yapısının nasıl çalıştığı anlatılacaktır. Aşağıdaki kod bloğunda deger_1 ve deger_2 şeklinde iki boolean değişkeni tanımlanmış ve kullanıcıların klavye aracılığıyla girdikleri boolean değerleri sırasıyla bu değişkenlere atanmıştır. Sonrasında mantıksal VEYA işlecinin bölümde verilen tanımına uygun olarak bu boolean değerlerinin mantıksal VEYA işlecinin sonucu belirlenmiş ve ekrana yazdırılmıştır. Örneğin kullanıcı deger_1 ve deger_2 değişkenleri için sırasıyla false ve true değerlerini girmiş olsun. Kod bloğundaki ilk koşul doğru olacağından, bu koşulun gövdesindeki kod çalıştırılacaktır. Bununla birlikte içteki if...else kontrol yapısındaki koşul doğru olmadığından, program koşulun gövdesindeki kodu pas geçecek ve alternatif gövdedeki kodu çalıştırarak ekrana "deger_1 || deger_2 ifadesinin boolean degeri DOGRU dur" cümlesini yazdıracaktır.

```
bool deger_1, deger_2;
cout << "İki bool degeri giriniz: ";
cin >> deger_1 >> deger_2;
if (deger_1 = = 0) {
        if (deger_2 = = 0) {
                cout << "deger_1 || deger_2 ifadesinin bool degeri YANLIS tir" << endl;
        }
        else {
                cout << "deger_1 || deger_2 ifadesinin bool degeri DOGRU dur" << endl;
        }
} else {
        cout << "deger_1 || deger_2 ifadesinin bool degeri DOGRU dur" << endl;
}
```

ARAŞTIRALIM

Yukarıdaki örneğe benzer olarak, iç içe if...else if yapısı kullanarak girilen iki adet boolean değerine mantıksal ve işlecini uygulayıp sonucu ekrana yazdıran bir kod geliştirin.

Şimdi de taş-kağıt-makas oyunu tasarlamak için kullanılabilecek bir kod bloğu üzerinden iç içe if... else kavramı pekiştirilecektir. Aşağıdaki kod bloğu seçimleri bilindiğinde yukarıda da aktarılan kurala göre oyunun kazananını belirlemektedir. Örneğin ilk oyuncu taşı ve ikinci oyuncu makası seçtiğinde, diğer bir ifadeyle oyuncu_1 ve oyuncu_2 string değişkenlerine atanan değerler sırasıyla "tas" ve "makas" olduğunda; kod bloğundaki ilk koşul doğru olacağından, bu koşulun gövdesindeki kod çalıştırılacaktır. Bununla birlikte içteki if...else kontrol yapısındaki ilk koşul doğru olmadığından, içteki kontrol yapısının ikinci koşulu test edilecektir. İçteki kontrol yapısının ikinci koşulu doğru olacağından bu koşulun gövdesindeki kod çalıştırılacak ve ekrana "birinci oyuncu kazandi" cümlesi yazdırılacaktır.

```
string oyuncu_1, oyuncu_2;
if (oyuncu_1 = = “tas”){
    if (oyuncu_2 = = “kagit”){
        cout << “ikinci oyuncu kazandi” << endl;
    }
    else if (oyuncu_2 = = “makas”){
        cout << “birinci oyuncu kazandi” << endl;
    }
    else {
        cout << “oyuncular berabere kaldilar” << endl;
    }
}
else if (oyuncu_1 = = “kagit”){
    if (oyuncu_2 = = “makas”){
        cout << “ikinci oyuncu kazandi” << endl;
    }
    else if (oyuncu_2 = = “tas”){
        cout << “birinci oyuncu kazandi” << endl;
    }
    else {
        cout << “oyuncular berabere kaldilar” << endl;
    }
}
else {
    if (oyuncu_2 = = “tas”){
        cout << “ikinci oyuncu kazandi” << endl;
    }
    else if (oyuncu_2 = = “kagit”){
        cout << “birinci oyuncu kazandi” << endl;
    }
    else {
        cout << “oyuncular berabere kaldilar” << endl;
    }
}
```

BÖLÜM ÖZETİ

Bu bölümde programın akışını kontrol etmeye imkan veren yapı türlerinden biri olan seçmeli kontrol yapılarından, ve bunun öncesinde seçmeli kontrol yapılarındaki koşulları oluşturabilmemiz için gerekli olan ilişkisel ve mantıksal işleçlerden bahsedilmiştir. İlişkisel işleçler, C++ programlama dilinde iki değişken ya da iki değer arasındaki ilişkiyi tanımlayan ya da koşullu ifadelerin bir parçası olarak bu ilişkiyi sınayan yapılardır. C++ programlama dilinde eşittir, eşit değildir, küçüktür, küçük eşittir, büyüktür ve büyük eşittir olmak üzere 6 adet ilişkisel işleç tanımlanmaktadır. İlişkisel işleçler üzerinden oluşturulmuş ilişkisel ifadelerin sonucu, eğer ifadedeki değerler ya da değişkenler arasında ilişkisel işlecin tanımladığı gibi bir ilişki varsa ifade true boolean değeri almakta, yoksa false boolean değeri almaktadır. Birden fazla durumun sınandığı daha karmaşık koşullu ifadelerin oluşturulabilmesi mantıksal işleçlerin kullanımıyla mümkün olmaktadır. C++ programlama dilinde DEĞİL, VE ve VEYA olmak üzere 3 adet mantıksal işleç tanımlanmaktadır. Bölüm içerisinde işlenenlerin farklı boolean değerleri için mantıksal işleçlerin aldığı boolean değerleri tablolar halinde gösterilmiştir. Seçmeli kontrol yapıları programın akışını belli koşullar üzerinden çatallandırmaya imkan veren yapılardır. C++ programlama dilinde if (tek seçim), if...else (ikili seçim) ve switch (çoklu seçim) olmak üzere üç farklı seçmeli kontrol yapısı tanımlanmıştır. If kontrol yapıları tek bir koşulun sınandığı yapılardır. Eğer koşul kısmındaki ifade doğru ise gövdedeki kod çalıştırılmakta, aksi durumda gövdedeki kod pas geçilmektedir. If yapısından farklı olarak if...else kontrol yapısı ise koşul yanlış olduğunda da belirli bir kod bloğunun çalıştırılabilmesine imkan vermektedir. If...else kontrol yapısında koşul sağlandığında gövdedeki kod, aksi durumda ise alternatif gövdedeki kod çalıştırılmaktadır. Birden fazla koşulun test edilmesi gerektiğinde ise if ve if...else kontrol yapıları yetersiz kalmaktadır. Bu gibi durumlarda çoklu seçime imkan veren if...else if veya switch kontrol yapıları tercih edilmelidir.

GÖZDEN GEÇİRELİM

1. **Aşağıdakilerden hangisi C++ programlama dilinde kullanılan ilişkisel işleçlerden biri değildir?**
   a. = =
   b. > =
   c. < =
   d. =
   e. >

2. **Aşağıdakilerden hangisi C++ programlama dilinde kullanılan mantıksal işleçlerden birisidir?**
   a. >=
   b. = =
   c. ||
   d. ! =
   e. <

3. **Aşağıda verilen kod bloğu çalıştırıldığında ekrana yazılacak değer aşağıdakilerden hangisi olacaktır?**

```
int sayi_1 = 1, sayi_2 = 2, sayi_3 = 3;
if ((3*sayi_1 = = sayi_2 + 1) && (sayi_3 < sayi_2
+ 2)){
cout << sayi_1 + sayi_3 << endl;
}
else {
cout << sayi_2 + sayi_3 << endl;
}
```

   a. 3
   b. 4
   c. 5
   d. 6
   e. 7

4. **Aşağıdakilerden hangisi C++ programlama dilinde kullanılan mantıksal VE işlecidir?**
   a. >=
   b. = >
   c. ||
   d. !&
   e. &&

5. **Aşağıda verilen kod bloğu çalıştırıldığında ekrana yazılacak değer aşağıdakilerden hangisi olacaktır?**

```
int sayi_1 = 2, sayi_2 = 3;
cout << ( (3*sayi_1 = = sayi_2*2) ? sayi_2
- sayi_1
: sayi_2 + sayi_1) << endl;
```

   a. 1
   b. 2
   c. 3
   d. 4
   e. 5

6. **Aşağıdaki anahtar kelimelerden hangisi switch kontrol yapısının bir parçası olarak kullanılmaktadır?**
   I. if
   II. case
   III. break
   IV. while
   V. default
   a. I ve III
   b. I ve V
   c. II ve IV
   d. II, III ve V
   e. II, IV ve V

7. Aşağıda verilen kod bloğu çalıştırıldığında ekrana yazılacak değer aşağıdakilerden hangisi olacaktır?

```
int sayi_1 = 8, sayi_2 = 6, sayi_3 = 7;
if (sayi_1 < sayi_2){
if (sayi_1 < sayi_3) {
cout << sayi_1 << endl;
  }
  else {
  cout << sayi_3 << endl;
  }
}
else {
if (sayi_2 < sayi_3) {
cout << sayi_2 << endl;
  }
else {
  cout << sayi_3 << endl;
}
}
```

a. 5
b. 6
c. 7
d. 8
e. 9

8. C++ programlama dilinde kullanılan ilişkisel küçük eşittir işleci aşağıdakilerden hangisidir?

a. <
b. = <
c. < =
d. < <
e. < = <

9. C++ programlama dilinde temp tamsayı değişkeninin 10'dan küçük pozitif bir tamsayı olup olmadığını kontrol eden mantıksal ifade aşağıdakilerden hangisinde doğru olarak verilmiştir?

a. temp > = 0 && temp < = 10
b. temp > = 0 && temp > 10
c. temp > 0 && temp < 10
d. temp < 0 && temp > 10
e. temp < 0 && temp < 10

10. Aşağıda verilen kod bloğu çalıştırıldığında ekrana yazılacak değer aşağıdakilerden hangisi olacaktır?

```
int sayi_1 = 4, sayi_2 = 5;
switch(sayi_1){
case 1: {
cout << sayi_2 - sayi_1 << endl;
break;
}
case 2: {
cout << 2*sayi_2 - sayi_1 << endl;
break;
}
case 3: {
cout << 3*sayi_2 - sayi_1 << endl;
break;
}
default: {
cout << 4*sayi_2 - sayi_1 << endl;
break;
   }
}
```

a. 1
b. 6
c. 11
d. 16
e. 21

Yanıt Anahtarı: 1-D, 2-C, 3-B, 4-E, 5-A, 6-D, 7-B, 8-C, 9-C, 10-D