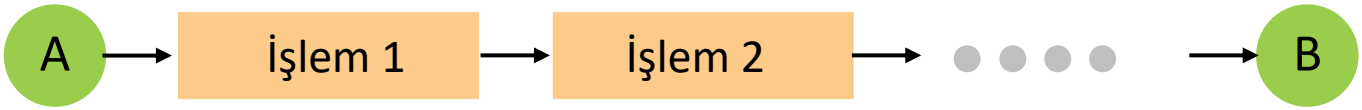
Akış Kontrolü

Genel olarak akış kontrolü, bir programın belirli durumlar veya koşullar doğrultusunda farklı işlemler yapmasını sağlar. Bu sayede program, girilen verilere veya karşılaşılan koşullara göre karar verebilir ve farklı adımlar izleyebilir. Akış kontrolü, işlemlerin sırasını yöneterek programın esnek ve dinamik bir şekilde çalışmasına olanak tanır, böylece belirli koşullara uygun çıktılar üretilir. Akış kontrolü işlemlerin gerçekleştirme şekline bağlı olarak;

- **Sıralı işlem yürütme**
- **Koşullu işlem yürütme**
- **Tekrarlayan işlem yürütme**

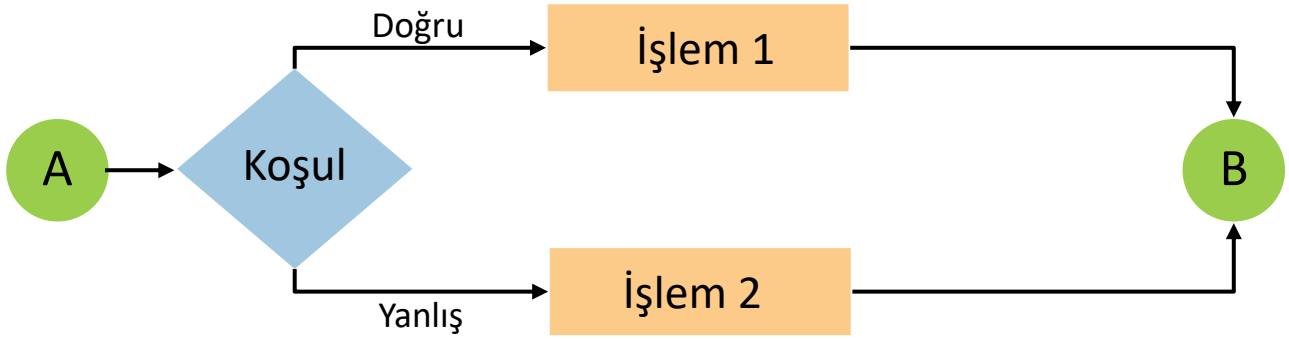
şeklinde 3 şekilde sınıflanabilir.

Sıralı işlem yürütme bir program akışının sıralı olarak ardışık işlemler ile gerçekleştirilmesidir. Bu zaten bir bilgisayar programın normal çalışma şeklidir. Her hangi bir koşul yada karar verme durumu söz konusu değildir. Program akışı belirli bir noktadan (A) başlar ve verilmiş olan komutlar sırasıyla işlenir. Daha sonra işlemlerin tamamlandığı noktada (B) program sonlandırılır.



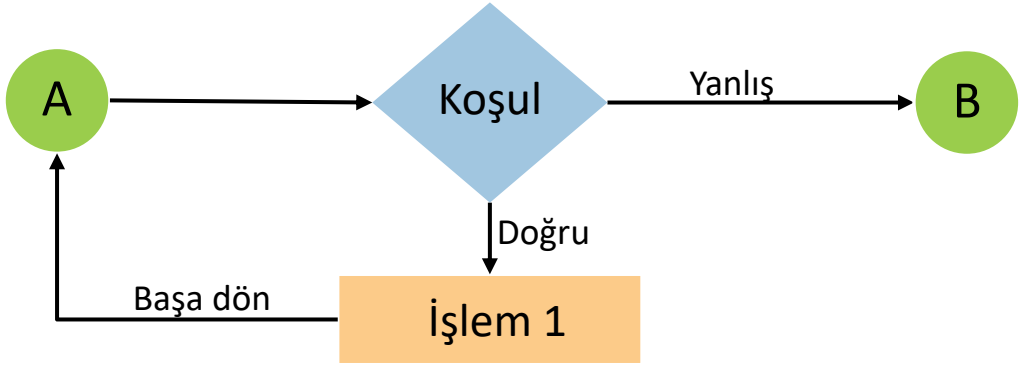
Koşullu işlem yürütme

Bu akış kontrolünde bir programın ilerlemesi belirli koşullara göre farklılıklar göstermektedir. Bu farklılıklar karar yapılarıyla (if-else) belirlenip her bir durum için ayrı işlemlerin yapılması sağlanır. A noktasından başlayan program akışı bir koşul ifadesi ile sınanır. Eğer koşul sağlanırsa **"işlem-1"** sağlanmıyorsa **"işlem-2"** yerine getirilir. Ardından sonra işlemlerin tamamlandığı noktada (B) program sonlandırılır.



Tekrarlayan işlem yürütme

Bu akış kontrolünde bir programın belirli işlemleri değişen parametrelere göre veya koşulların sağlanıp sağlanmadığına göre tekrarlamasıdır. A noktasından başlayan program akışı bir koşul ifadesi ile sınanır. Eğer koşul sağlanırsa **"işlem-1"** yerine getirilir ve tekrar A noktasına geri dönülerek koşul tekrar sınanır. Eğer koşul sağlanmıyorsa işlemlerin tamamlandığı noktada (B) program sonlandırılır.



Karar Yapıları

C'de kullanılan karar yapıları **"if"**, **"if-else"**, **"else if"** ve **switch** komutlarıdır. Bunların içinde en önemlisi **"if-else"** yapısıdır. Bu yapı, belirli bir koşulun doğru veya yanlış olmasına göre programın ne yapacağını belirler. Örneğin, "eğer bir sayı 10'dan büyükse **'büyük'** yaz, değilse **'küçük'** yaz" gibi bir durumu yönetmek için **"if-else"** kullanılır. Koşullar, mantıksal ifadelerle (örneğin, `==`, `>`, `<`) belirlenir ve bu ifadeler doğru olduğunda ilgili kod bloğu çalıştırılır. C'de ayrıca **"else if"** anahtar kelimesiyle çoklu koşulların kontrol edilmesini sağlar. Böylece, birden fazla durum kontrol edilebilir.

"if" komutu tek başına bir koşulun sağlandığı durumlar ile kullanılırken, **"if-else"** çifti bir koşulun sağlandığı ve sağlanmadığı şeklinde iki farklı durumlar için kullanılır. **"if-else if"** komutu ise 2'den fazla koşulların sağlandığı durumlar için kullanılır ve iç-içe geçmiş birçok koşulların kontrolü için de kullanır. Bu durumların genel olarak yazım düzenleri şöyledir:

Tek koşullu durum

```
if (koşul){

    işlem;

}
```

İki koşullu durum

```
if (koşul){

    işlem;

} else {

    işlem;

}
```

Çok koşullu durum

```
if (koşul1){

    işlem;

} else if (koşul2){

    işlem;

} else {

    işlem;
}
```

"if" ve **"else if"** komutlarındaki koşul durumları karşılaştırma operatörleri ile kontrol edilir. Karşılaştırma operatörleriyle bir değişken belirli sabitlerle kıyaslandığı gibi farklı değişkenler de birbirleriyle kıyaslanabilir. Eğer "if-else" koşullarında sonra 1'den fazla işlem yapılmayacaksa parantez "{ }" bloklarının konulması gerekli değildir.

Karşılaştırma ve Mantık Operatörleri (Hatırlatma)

Program içerisinde bir değişkenin değerini başka bir değişkenin değeri veya bir sabitle karşılaştırılmasını sağlayan operatörlere ''ilişkisel veya karşılaştırma operatör" adı verilir. Karşılaştırma durumuna göre sonuç "1" veya "0" ya da "True/False" yani "Doğru/Yanlış" olarak algılanır. Karşılaştırma işleminde istenen şart sağlanmıyorsa sonuç her zaman "yanlış" yani "0", şart sağlanıyorsa sonuç "doğru" yani "1" olur. Karşılaştırma operatörleri;

Operatör	Örnek	Anlamı
>	a > b	a, b'den büyük
<	a < b	a, b'den küçük
>=	a >= 7	a, b'den büyük veya b'ye eşit
<=	a <= b	a, b'den küçük veya b'ye eşit
==	a == b	a, b'ye eşit
!=	a != b	a, b'ye eşit değil veya b'den farklı

Karşılaştırma operatörleri her hangi bir koşulun sınanması amacıyla kullanıldıkları zaman, karşılaştırma sonucu doğru (true) ya da yanlış (false) olarak ifade edilir. Koşul doğruysa olumlu varsayılır ve buna göre yapılması istenenler işlemler yerine getirilir. Yanlış ise olumsuz varsayılır ve buna göre ya başka işlemler yapılır ya da hiçbir işlem yapılmaz atlanır.

Mantık operatörleri "VE / VEYA" her zaman iki koşulu karşılaştırmak için kullanılır. Yani "A && B" ya da "A || B" şeklinde kullanılırlar. Ayrıca gruplama suretiyle birden fazla karşılaştırma yapmak da mümkündür.

1. **"&& - and"** : Tüm koşulların doğru olması durumunda "True" döner. Aksi halde "False" döner.
2. **"|| - or"** : Koşullardan en az birinin doğru olması durumunda "True" döner. İki koşul da yanlışsa "False" döner.
3. **"! - not"** : Mantıksal değeri tersine çevirir. "True" olan bir ifadeyi "False", "False" olanı "True" yapar.

printf ve scanf fonksiyonlarında yer tutucular (Hatırlatma)

"printf()" ve "scanf()" fonksiyonlarında farklı veri türleri için kullanılan yer tutucu sembollerin listesi.

Yer Tutucular	Anlamı
%d	int türünü tamsayı olarak yazar
%ld	long türünü tamsayı olarak yazar
%x	işaretsiz int türünü hexadecimal sistemde yazar
%X	işaretsiz int türünü hexadecimal sistemde yazar (semboller büyük harfle)
%lx	işaretsiz long türünü hexadecimal sistemde yazar
%u	işaretsiz int türünü ondalık sistemde yazar
%o	işaretsiz int türünü oktal sistemde yazar
%f	float ve double türlerini kesirli olarak yazar
%lf	double türünü kesirli olarak yazar
%e	gerçek sayıları üstel biçimde yazar
%c	char veya int türünü karakter görüntüsü tek harf olarak yazdırır
%s	metin dizisini yazar
%lf	long double türünü kesirli olarak yazar

short türü bir sayı yazarken d, o, u veya x karakterlerinden önce **"h"** karakteri kullanılır, long türü bir için ise **"l"** karakteri kullanılır. Ayrıca **%%** kullanımı **"%"** karakterinin yazılacağını belirtir. Tam sayıları veya metinleri formatlı şekilde göstermek için gösterilecek rakam veya harf sayısı **"%d"** yerine **"%nd"** kullanarak belirtebiliriz. Kesirli ve üstel sayılarda **"%f/e"** yerine **"%n.mf/e"** kullanılır. Burada **"n"** sayısı, tam kısmındaki rakam sayısını **"m"** kesirli kısımdaki rakam sayısını gösterir

if koşul örnekleri

"Eğer n sayısının değeri 10'dan büyükse, 'sayı 10'dan büyüktür' mesajı yazılsın" şeklinde bir ifadeyi dikkate alalım. Burada koşulumuz sayısının 10'dan büyük olmasıdır. Bu durumu C'de ifade edersek,

```
if (n > 10)
    printf("sayı 10'dan büyüktür");
```

şeklinde olacaktır. Burada "if"den sonra ";" olmadığına dikkat edilmelidir. Kod editöründe yeni dosya oluşturup aşağıdaki kodları yazalım ve **sayikontrol.c** olarak kaydedelim.

```
printf("Bir sayı giriniz: ");
scanf("%d", &sayi);

if (sayi > 10)
    printf("Girdiğiniz sayı 10'dan büyüktür!");

if (sayi < 10)
    printf("Girdiğiniz sayı 10'dan küçüktür!");

if (sayi == 10)
    printf("Girdiğiniz sayı 10'dur!");
```

```
C kodunun çalışması için;

#include <stdio.h>
void main(){}
int sayi;

tanımlamaları yapılmalıdır
```

Gördüğünüz gibi, art arda üç adet **"if"** bloğu kullandık. Bu kodlara göre,

- eğer kullanıcının girdiği sayı 10'dan büyükse, ilk **"if"** bloğu işletilecek,
- eğer sayı 10'dan küçükse ikinci **"if"** bloğu işletilecek
- eğer sayı 10'a eşit ise üçüncü **"if"** bloğu işletilecektir.

Bu programı çalıştırıp farklı sayılar girerek test edelim.

VSCode terminalinde Türkçe karakterlerin doğru görünmesi için "chcp 65001" komutu girilmelidir.

if-else Koşul örnekleri

Şimdide bir sisteme şifre ile giriş durumunu dikkate alalım. Burada kullanıcı bir şifre girecek ve doğru ise "Sisteme hoş geldiniz" mesajı aksi durumda "Yanlış şifre girdiniz" mesajı iletilecek olsun. Kod editöründe yeni dosya oluşturup aşağıdaki kodları yazalım ve **sistem.c** olarak kaydedelim.

```
printf("Üniversite kütüphane hizmetine\n\
Hoş geldiniz. Yalnız hizmetimizden\n\
yararlanmak için önce sisteme giriş\n\
yapmalısınız.\n\n\
Şifre: ");

scanf("%d", &sifre);

if (sifre == 12345678)
    printf("Sisteme hoş geldiniz!");

else
    printf("Yanlış şifre girdiniz!");
```

Programın başında "\n" ve "\" sembollerinden yararlanarak uzun bir metni kullanıcıya gösterdik. Bu bölümü, kendiniz göze hoş gelecek bir şekilde süsleyebilirsiniz de. Eğer kullanıcı, kendisine parola sorulduğunda cevap olarak "12345678" yazarsa kullanıcı sisteme giriş yapar ve kendisine "Sisteme hoş geldiniz" mesajı yazılır, aksi durumda "Yanlış şifre girdiniz" mesajı yazılır.

İf-else koşul örnekleri

Klavyeden kullanıcının girdiği bir metinin, arzu edilen metin olup olmadığını kontrol eden bir program yazmaya çalışalım. Eğitici bir oyun yazmaya çalışalım, oyunumuzda bir tanım ve karşılığında bir kelime girilsin. Girilen kelime doğru ise bunu kullanıcıya bildirelim. Kod editöründe yeni dosya oluşturup aşağıdaki kodları yazalım ve **cevaptest.c** olarak kaydedelim.

```
printf("Çorum, Sivas ve Kayseri ile komşu olan ilimiz hangisidir?");
printf("Cevabınız: ");
scanf("%s", &girilen);

if (strcmp(girilen, "yozgat")==0)
    print("Tebrikler cevabınız doğru");
```

```
C kodunun çalışması için;

#include <stdio.h>
#include <string.h>
void main(){}
char girilen[10];

tanımlamaları yapılmalıdır
```

Burada metin karşılaştırma işlemi **"strcmp"** komutu ile yapılır. Program çalıştırıldığında çıktılar şöyle görünecektir.

```
Çorum, Sivas ve Kayseri ile komşu olan ilimiz hangisidir?
Cevabınız: yozgat
Tebrikler cevabınız doğru
```

Burada dikkat edilmesi gereken bir durum vardır. Eğer kullanıcı şehir isminin ilk harfini büyük olarak girerse cevap uygun olamayacaktır. Çünkü Python büyük-küçük harf ayırımı yapmaktadır, yani "yozgat" ile "Yozgat" aynı değildir. Bu nedenle "if" satırı şu şekilde düzenlenmelidir.

```
if (strcmp(girilen, "yozgat")==0 | strcmp(girilen, "Yozgat")==0)
```

Bu şekilde istenilen doğru cevap her iki hal için de kabul edilmiş olacaktır.

İf-else koşul örnekleri

Şimdi önceki örnekte verilen cevabın uygun olmadığı durumu dikkate alalım. Kod editöründe **cevaptest.c** kodunu aşağıdaki gibi düzenleyelim ve **cevaptest2.c** olarak kaydedelim.

```
printf("Çorum, Sivas ve Kayseri ile komşu olan ilimiz hangisidir?\n");
printf("Cevabınız: ");
scanf("%s", &girilen);

if (strcmp(girilen, "yozgat")==0 | strcmp(girilen, "Yozgat")==0)
    printf("Tebrikler cevabınız doğru.");

else
    printf("Üzgünüm cevabınız doğru değil. Yeniden deneyin");
```

Program çalıştırılıp cevap olarak "bursa" verildiğinde çıktı aşağıdaki şekilde görünecektir.

```
Çorum, Sivas ve Kayseri ile komşu olan ilimiz hangisidir?
Cevabınız: bursa
Üzgünüm cevabınız doğru değil. Yeniden deneyin
```

Ancak cevap olarak "yozgat" ya da "Yozgat" verildiğinde ise çıktı "Tebrikler cevabınız doğru." olacaktır.

"if" - "else if" koşul örnekleri

Bir kullanıcının yaşını alıp yaşına göre belirli yorumlar yapılan bir durumunu dikkate alalım. Kod editöründe yeni bir dosya oluşturup aşağıdaki kodları ayrı ayrı yazalım ve **yaskontrol1.c** ve **yaskontrol2.c** olarak kaydedelim.

```c
printf("Yaşınız: ");
scanf("%d", &yas);

if (yas == 18)
    printf("18 iyidir!");

else if (yas < 0)
    printf("Yok canım, daha neler!...");

else if (yas < 18)
    printf("Genç bir kardeşimizsin!");

else if (yas > 18)
    printf("Eh, artık büyüdün sayılır!");
```

```c
printf("Yaşınız: ");
scanf("%d", &yas);

if (yas == 18)
    print("18 iyidir!");

if (yas < 0)
    print("Yok canım, daha neler!...");

if (yas < 18)
    print("Genç bir kardeşimizsin!");

if (yas > 18)
    print("Eh, artık büyüdün sayılır!");
```

Bu iki programın da aynı işlevi gördüğünü düşünebilirsiniz. Ancak ilk bakışta pek belli olmasa da, aslında yukarıdaki iki program birbirinden farklı davranacaktır. Örneğin ikinci programda eğer kullanıcı eksi değerli bir sayı girerse hem **"if (yaş < 0)"** bloğu, hem de **"if (yaş < 18)"** bloğu çalışacaktır. Bunu görmek için yukarıdaki programı çalıştırıp, cevap olarak eksi değerli bir sayı verelim.

"if" – "else if" koşulu diğer bir örnek

Bir kullanıcının girdiği ders puanını kontrol eden ve puana göre harf notunu yazan bir durumu dikkate alalım. Kod editöründen bir dosya açıp aşağıdaki kodları yazalım ve **notkontrol.c** olarak kaydedelim.

```
printf("Öğrenci puanının girin : ");
scanf("%d", &puan);

if (puan > 85)
    karne_notu = 'A';

else if (puan > 70)
    karne_notu = 'B';

else if (puan > 60)
    karne_notu = 'C';

else if (puan > 50)
    karne_notu = 'D';

else
    karne_notu = 'F';

printf("Karne notu: %c", karne_notu);
```

```
printf(" Öğrenci puanının girin : ");
scanf("%d", &puan);

if (puan > 85){karne_notu = 'A';}
else if (puan > 70){karne_notu = 'B';}
else if (puan > 60){karne_notu = 'C';}
else if (puan > 50){karne_notu = 'D';}
else{karne_notu = 'F';}

printf("Karne notu: %c", karne_notu);
```

Görüldüğü üzere harf notları büyükten küçüğe göre sıralanarak verilmiştir. Bu sadece işlem kolaylığı içindir. Belirli aralıklar olarak yazılması durumda daha fazla kod yazımı söz konusu olacaktır.

```
if ((puan > 70) && (puan <= 85)) karne_notu = 'B'
if ((puan > 60) && (puan <= 70)) karne_notu = 'C'
if ((puan > 50) && (puan <= 60)) karne_notu = 'D'
```

"if" – "else if" koşulu diğer bir örnek

Kullanıcıdan 0 ile 50 arasında bir sayı alıp bunun hangi 10'nun katları (0-10,10-20, gibi) arasında olduğu yazan bir programı dikkate alalım. Kod editöründen bir dosya açıp aşağıdaki kodları yazalım ve **onkat.c** olarak kaydedelim.

```
printf("0 ila 50 arası bir sayı girin: ");
scanf("%d", &sayi);

if ((sayi < 0) || (sayi > 50)){

    printf("Girilen sayı 0 ila 50 arasında değil!");

}else if (sayi < 10){

    printf("Girilen sayı 0\'dan büyük 10\'dan küçüktür");

}else if (sayi < 20){

    printf("Girilen sayı 10\'dan büyük 20\'den küçüktür");

}else if (sayi < 30){

    printf("Girilen sayı 20\'den büyük 30\'dan küçüktür");

}else if (sayi < 40){

    printf("Girilen sayı 30\'dan büyük 40\'tan küçüktür");

}else{

     printf("Girilen sayı 40\'tan büyüktür");
}
```

Kodu derledikten sonra farklı değerler vererek test edelim. Eğer sayı olarak 60 değerini girersek program şu mesajı verecektir.

```
Girilen sayı 0 ila 50 arasında değil!
```

İç-içe geçmiş (nested) if-else-else if yapıları

İç içe "if-else-else if" yapıları, bir koşulun içinde başka bir koşulu kontrol etmek için kullanılır. Bu yapılar, daha karmaşık karar verme süreçlerini yönetmek veya koşullara bağlı olarak adım adım işlemler gerçekleştirmek gerektiğinde tercih edilir. İç içe koşul yapıları, dış koşul doğruysa, iç koşulların da kontrol edilerek programın belirli bir yöne yönlendirilmesini sağlar. Bu iç-içe yapıların kullanım amaçları şöyle özetlenebilir:

1. **Karmaşık Karar Verme Süreçleri:** İç içe yapılar, bir ana koşul sağlandığında başka koşulları da kontrol ederek, daha ince ayrıntılara dayalı kararlar almayı sağlar.
2. **Adım Adım Kontrol:** Bazen bir durumu aşama aşama kontrol etmek gerekebilir. Bu aşamalı kontrol iç içe if yapıları ile kolayca sağlanabilir.
3. **Daha Detaylı Kontroller:** Ana koşulun doğru olup olmadığını kontrol ettikten sonra, alt koşullarla daha hassas kontroller yapabilirsiniz.

İç-içe yapılar karşılaştırılan koşulların sayısına bağlı olarak oldukça karmaşık yapılarda olabilirler. Bunun en basit hali olan 3 koşullu bir durumu dikkate alırsak iç-içe yapı şöyle görünecektir.

```
if (koşul_1){
    işlemler_1_1;
    if (koşul_2){
        işlemler_2_1;
    }else{
        işlemler_2_2;
    }
}else{
    if (koşul_3){
        işlemler_3_1;
    }else{
        işlemler_3_2;
    }
}
```

Dikkat Edilmesi Gerekenler

- **Kodun okunabilirliği**: İç içe yapılar mantıksal olarak faydalı olsa da, aşırı kullanımı kodu karmaşık hale getirebilir. Bu nedenle, mümkün olduğunca kodun anlaşılır ve temiz olmasına dikkat edilmelidir.
- **Parantez bloğu:** C'de bloklar parantez "{}" sembolleri tanımlandığı için, iç içe if yapılarında bunlara dikkat etmek çok önemlidir. Hatalı bloklar beklenmedik sonuçlara yol açabilir.

İç-içe if-else-elif yapısı örneği

Bir öğrencinin sınıf geçme notuna göre başarı durumunu değerlendirmek istediğimizi varsayalım. Notu 50'nin üzerinde olan öğrencilere **"geçer not"** ile, notu 80'ın üzerinde olan öğrencilere **"teşekkür belgesi"** ile, notu 90'ın üzerinde olan öğrencileri **"onur belgesi"** ile mezun olarak değerlendirmek istiyoruz.

```
int geçme_notu = 95;

if (geçme_notu >= 50){

    printf("Sınıfı Geçtiniz.\n");

    if (geçme_notu >= 90){

         printf("Onur Belgesi kazandınız.\n");

    }else if (geçme_notu >= 80){

         printf("Teşekkür Belgesi kazandınız.\n");

    }else{
         printf("Tebrikler.\n");
    }

}else{
    printf("Maalesef sınıfta kaldınız...\n");
}
```

- **İlk if bloğu:** "geçme_notu >= 50" koşulu kontrol edilir. Eğer doğruysa (ki burada 95 olduğu için doğru), öğrenci geçmiştir.

- **İkinci if bloğu (iç içe if):** "geçme_ notu >= 90" koşulu tekrar kontrol edilir. Eğer doğruysa (bu örnekte doğru), öğrenci onur belgesi kazanmıştır. Eğer bu koşul doğru değilse, "else if" ve "else" blokları kontrol edilir.

- Eğer ilk **"if"** bloğundaki koşul sağlanmamış olsaydı, yani notu 50'nin altında olsaydı, program direk **"else"** bloğuna geçer ve "Maalesef sınıfta kaldınız" mesajını yazdırırdı.

İç-içe if-else-elif yapısı diğer bir örnek

Girilen her hangi bir tamsayının 2'ye ve 3'e bölünebilme koşullarını kontrol ederek sayının her ikisine ya da sadece birine bölünebildiği durumları kontrol edelim. Daha bu durumları belirten mesajları hazırlayalım. İlk adımda kullanıcıdan bir sayı girişi yapıldığını varsayıyoruz.

```
printf("Bir sayı girin: ");
scanf("%d", &sayi);

if (sayi%2 == 0){

    if (sayi%3 == 0){
         printf("%d sayısı hem 2\'ye hem de 3\'e bölünmektedir.\n", sayi)

    }else{
          printf("%d sayısı sadece 2\'e bölünmektedir.\n", sayi);

    }

}else{

    if (sayi%3 == 0){
         printf("%d sayısı sadece 3\'ye bölünmektedir.\n", sayi);

    }else{
          printf("%d sayısı 2\'ye ve 3\'e bölünmeMEktedir.\n", sayi);

    }
}
```

Programı çalıştırıp farklı sayı girişleri ile test edelim. İlk olarak 8 sayısını daha sonra sırasıyla 15, 12 ve 5 sayılarını girdiğimiz zaman yandaki çıktıları alacağız.

```
Bir sayı giriniz: 8
    8 sayısı sadece 2'e bölünmektedir.
Bir sayı giriniz: 15
    15 sayısı sadece 3'e bölünmektedir.
Bir sayı giriniz: 12
    12 sayısı hem 2'ye hem de 3'e bölünmektedir.
Bir sayı giriniz: 5
    5 sayısı 2'ye ve 3'e bölünmeMEktedir.
```

Tek satır if-else-else if yapısı, KOŞULLU OPERATÖR "?:" kullanımı

Bazı "if" koşul durumlarında işlemler çok kısa olup sadece bazı değişkenlerin belirli değerler alması gerekebilir. Bu durumda "if-else-else if" yapısını alt alta satırlar şeklinde yazmak yerine tek satıra yazılabilir. Ancak bu yazım düzeni 2'den fazla koşulların olduğu durumlarda kodun okunabilirliğini ve anlaşılmasını zorlaştırabilmektedir.

Örneğin a ve b gibi iki sayının büyüklükleri kıyas eden tek koşullu bir durum için;

```
if (a > b){
    print("%d, %d'den büyüktür.", a, b)
}
```

yerine aşağıdaki ifadeyi kullanmak kod kısaltma açısından daha uygun olur.

```
if (a > b) print("%d, %d'den büyüktür.", a, b)
```

2 koşullu bir örnek olarak, bir sayının çift ya da tek sayı olduğunu belirtecek durumu tek satır "?:" yapısı ile ifade etmek daha kolay olabilir. Burada "?" sembolünden sonra koşulun doğru sonucu ":" sembolünden sonra ise koşulun yanlış işlemi yazılır.


```
a = 10
durum = (a%2 == 0) ? "Çift sayı" : "Tek sayı";
printf("%d sayısı %s'dır.", a, durum)
```

normal yazım düzeni

```
if (a%2 == 0) durum = "Çift sayı";
else durum = "Tek sayı";
```

Tek satır if-else-else if yapısı, KOŞULLU OPERATÖR "?:" kullanımı

Eğer koşul sayısının 2'den fazla olduğu durumlarda kod yazımı tek satıra sığdırılsa bile kodu anlaşılması oldukça zor olabilmektedir. Örneğin verilen bir yaş kriterine göre kişinin "bebek", "genç" ya da "çocuk" olduğu bir durumu dikkate alalım.

normal yazım düzeni

```
if (yas >= 18) durum = "Genç" ;
else if (yas < 5) durum = "Bebek";
else durum = "Çocuk";
```

```
yas = 10
durum = (yas >= 18) ? "Genç" : ((yas < 5) ? "Bebek" : "Çocuk");
printf("Kişi %s", durum);
```

Görüldüğü üzere tek satır "?:" yazım düzeninin ilk bakışta anlaşılması, normal "if-else-else if" yazım düzenine göre biraz zordur. Bu bakımdan dikkatli bir şekilde ayarlanması gereklidir. Aksi takdirde istenilen sonuçlar doğru bir şekilde ifade edilmeyecektir.

Diğer bir örnek

Hava sıcaklığının "soğuk", "sıcak" ve "ılık" olarak ifade edilebileceği bir durumu dikkate alalım. Hava sıcaklığının 30 ile 20 arasında "ılık", üstünün "sıcak", altının da "soğuk" olduğunu kabul edelim. Bu durumu gösteren kod şöyle olacaktır.

```
#include <stdio.h>

void main() {
int sicaklik = 25;
char* hava;     // char hava[5];

hava = (sicaklik > 30) ? "sıcak" : (sicaklik > 20) ? "ılık" : "soğuk";
printf("%s\n", hava);
}
```

switch-case komutu

C dilinde **"switch-case"** yapısı, belirli bir değere veya duruma göre farklı kod bloklarının çalıştırılmasını sağlayan bir akış kontrol mekanizmasıdır. **"switch-case"** içinde akış kontrolünün daha verimli kullanılması amacıyla **"break"** ve **"default"** kelimeleri kullanılır. **"break"** ifadesi kontrol akışının hemen bitirilmesi amacıyla, **"default"** ifadesi ise hiçbir seçeneğin sağlanmadığı durumlar için kullanılmaktadır.

Genel yazım düzeni

```
switch değişken_ismi{

   case deger_1:

       ifade 1

    case deger_2:

       ifade 2

    ...

    case deger_n:

       ifade n

    default:

       ifade genel
}
```

Örnek kullanım

```
printf("Gün numarası girin: ");
scanf("%d", &gun);

switch (gun){
    case 1: printf("Pazartesi\n"); break;
    case 2: printf("Salı\n"); break;
    case 3: printf("Çarşamba\n"); break;
    case 4: printf("Perşembe\n"); break;
    case 5: printf("Cuma\n"); break;
    case 6: printf("Cumartesi\n"); break;
    case 7: printf("Pazar\n"); break;
    default: printf("Yanlış gün numarası");
}

// 3 için Çarşamba
// 6 için Cumartesi
// 0 için Yanlış gün numarası
```

"case" içinde çoklu değerler için işlem yapılacağı zaman "case" ifadeleri ":" ile birlikte "case a: case b: case c:" şeklinde yan yana yazılabilirler. En önemlisi **"swith-case"** ifadesinde sadece tamsayı türünden değişmezler kullanılmaktadır. Kesirle sayılar ve matematiksel işlemler kullanılamaz.

"switch-case" için geçerli/geçersiz durumlar

Bilindiği gibi değişmez ifadeleri, derleme aşamasında derleyici tarafından net sayısal değerlere dönüştürülebilir:

```
case 1 + 3:    /* Geçerli */
```

mümkün çünkü 1 + 3 sonucu tamsayı değişmez ifadesidir. Ancak,

```
case x + 5:    /* Geçersiz */
```

çünkü değişmez ifadesi değil. Derleyici, derleme aşamasında sayısal bir değer hesaplayamaz.

```
case 'a':      /* Geçerli */
```

Yukarıdaki **"case"** ifadesi geçerlidir. 'a' bir karakter değişmezidir. **"case"** ifadesi tamsayı türünden bir değişmez ifadesidir. Buna benzer olarak iki tek tırnak içinde tek karakter olarak belirli sembol ve harflerin tümü "case" ifadesinde kullanılabilir. Çünkü her bir sembolün bir ondalık değeri bulunmaktadır ve bu tamsayı değişmezidir.

```
case 3.5:
```

Yukarıdaki **"case"** ifadesi geçersizdir. 3.5 bir gerçek sayı değişmezidir, tamsayı değişmezi değildir.

```
case 'a' && 'b':
case 'a' || 'b':
```

Şeklindeki "case" ifadeleri de geçersizidir. Çünkü mantıksal operatörler ile işlem yapılamaz.

switch-case kullanımı diğer bir örnek

Aşağıdaki kod **"case"** deyimlerinin nasıl birleştirileceğini gösterir. İlk case bloğunda ":" noktadan sonra hiçbir ifade yoktur. Bur dikkat edilecek nokta, birleştirilecek "case" blokları arasına "break" ya da başka bir ifadenin konulmamasıdır. Boş "case" blokları kodun biraz anlamsız görünmektedir. Böyle durumlar için "if-else-else if" bloklarının kullanılması daha uygun olmaktadır.

```
#include <stdio.h>

void main(){
    int kod;
    char* yetki;

    printf("0 : Yönetici\n\
       1 : İdareci\n\
       2 : Misafir\n\
       3 : Diğer\n\n\
       Erişim kodunu girin:");

    scanf("%d", &kod);

    switch(kod){
        case 0: case 1:
            yetki = "Tam erişim";
            break;

        case 2:
            yetki = "Sınırlı erişim";
            break;

        default: yetki = "Erişim yok";
    }

    printf("%s",yetki);
}
```

İç-içe "switch-case" kullanımı

Tıpkı iç içe **"if-else"** yapıları gibi, iç içe "switch-case" yapıları da kullanılabilir. Bir "switch-case" yapısında her hangi "case" yapısı içinde başka bir "switch-case" yapısı konulabilir.

Genel yazım düzeni

```
switch (değişken_1){

   case deger_1:

       switch (değişken_2){

           case deger_a:
               ifade_a;
               break;

           case deger_b:
               ifade_b;
               break;
       }

   case deger_2:

        switch (değişken_3){

           case deger_c:
               ifade_c;
               break;

           case deger_d:
               ifade_2;
               break;
       }
}
```

Örnek bir kod

```
int x = 1, y = 2;

switch (x) {          // dış switch
    case 1:

        switch (y) {       // iç switch
           case 2:
               printf("Seçim 2\'dir");
               break;

           case 3:
               printf("Seçim 3\'tür")
               break;
        }
        break;

    case 4:
        printf("Seçim 4\'tür");
        break;

    case 5:
        printf("Seçim 5\'tir");
        break;

    default:
        printf("Seçim 1, 2 3, 4, ya da 5 değildir");

}
```