# 5

# Program Akışının Denetimi

Bir programın akışı komutların yazıldığı sırayı izler. Ama çoğunlukla, bu akışı yönlendirmek gerekir. Bu iş için denetim yapılarını kullanırız. Bunlar üç gruba ayrılabilir: *Bloklar*, *Yönlendiriciler* ve *Döngüler*. Bilgisayarın karmaşık işleri yapmasını sağlayan bu yapılar altı tanedir:

1. blok,
2. if ... else ... ve *else if* yönlendirmeleri
3. switch yönlendirmesi
4. Koşullu yönlendirme (koşullu üçlem)
5. for döngüsü
6. while ... döngüsü
7. do ... while ... döngüsü

Bu yapılardan her birisi tek bir deyimmiş gibi iş yapar, ama genellikle her birisi birden çok deyim içeren birer yapıdır. Bunların her birisini örneklerle açıklayacağız. Bu bölümde yönlendiriciler dediğimiz ilk dört yapıyı, sonraki bölümde ise döngüler dediğimiz son üç yapıyı ele alacağız.

## 5.1 Blok

Bir arada yürütülmesi istenen ve { } parantezi içine yazılan deyimlerden oluşan program parçasıdır. Sözdizimi şöyledir:

```
{
    deyimler
}
```

{ } bloku içindeki deyimler gerektiği sayıda C deyimlerinden oluşur. Hiç deyim içermeyen bloklara *boş blok* denir. *İçiçe* bloklar olabilir. Bir fonksiyonun gövdesi bir bloktur. Yönlendiriciler birer bloktur. Blok içindeki deyimler, yalın C deyimleri olabileceği gibi, yapısal deyimler de olabilir. Program akışı (akış denetimi) bir blok içine girerse, o bloktaki bütün deyimleri sırayla işler. Ancak, gerektiğinde, blok içindeki deyimler bitmeden, program akışı blok dışına alınabilir.

**Uyarı 5.1.**

`gcc` derleyicisinin *ascii* dışındaki Türkçe karakterleri metin içinde yazabilmesi için `locale.h` başlık fonksiyonu çağrılır ve `main()` fonksiyonunun gövdesine,

```
setlocale(LC_ALL,"");
```

deyimi konulur.

### Blok Örnekleri

Program 5.1, $x$ ile $y$ değişkenlerinin değerlerini takas eder. main() fonksiyonunun gövdesi en dış { } blokdur. Onun içinde *takas* işlemini yapan blok yer alır. Görüldüğü gibi bu örnekte iki tane içiçe blok vardır.

**Program 5.1.**

```
#include <stdio.h>
#include <locale.h>

int main(void)
{
  setlocale(LC_ALL,"");

  int x = 10, y = 20, z;
  {
    printf("Takastan önce : \n");
    printf("x = %d ; y = %d \n "   , x, y);
    z = x;
    x = y;
    y = z;
  }
  printf("Takastan sonra : \n");
  printf("x = %d ; y = %d \n"    , x, y);
}
```

```
/**
  Takastan önce :
```

```
    x = 10 ; y = 20
  Takastan sonra :
    x = 20 ; y = 10
*/
```

Blok içinde bildirimi yapılan bir değişken, o blok için bir *yerel* değişkendir. Program 5.1'deki $x, y, z$ değişkenleri *main()* metodunun yerel değişkenleridir. Eğer 5.satır iç bloka alınırsa; yani $x, y, z$ değişkenleri iç blokun yerel değişkenleri yapılırsa, takas işlemi olur, ama 14.satır çalışmaz. Çünkü blok dışından blok içindeki değişkenlere erişilemez. Onlara erişmek için 15.satırın da iç blokun içine alınması gerekir.

Özetlersek, *yerel* değişkene kendi bloku dışından erişilemez; ona ancak kendi bloku içindeki deyimler erişebilir.

## 5.2 Kapsanma Alanı (scope)

Blokun işi bitince, blok ile birlikte *yerel* değişkene ana bellekte ayrılan adres yok olur. Bu nedenle, bir değişkenin *kapsanma alanı* (erişilebirlik bölgesi, scope) o değişkenin tanımlandığı bloktur.

## 5.3 Yönlendiriciler

(Saptırma, Dallandırma)

Aksi istenmemişse, program, komutların yazıldığı sırada işler; yani normal durumda programın akışı (deyimlerin işleniş sırası) doğrusaldır. Ama oluşan koşullara bağlı olarak, program akışını başka deyimlere yönlendirmek gerekebilir. Bu işi yapan yönlendirici yapılar vardır. Yönlendiriciler, program akışını, oluşan koşullara göre istenen yöne saptıran (dallandıran) denetim yapılarıdır. Bunlar da kendi içlerinde türlere ayrılabilir: *if*, *if-else*, *else if*, *switch* ve *koşullu üçlem*. Bu yapılar ard arda gelebilir ya da içiçe yuvalanabilirler. Böylece daha karmaşık yönlendiriciler oluşur. Ancak, karmaşık yönlendiriciler, çözümlenerek, her bir parçası yukarıdaki temel yönlendiricilerden birisine indirgenebilir.

### 5.3.1 Yalın if Yapısı

Bazı deyimlerin işlemesini, ancak belirli koşulların sağlanması durumunda isteyebiliriz. Bu durumda if yönlendirmesini kullanırız. Bu yönlendirmenin sözdizimi aşağıdaki gibidir.

```
if ( mantıksal_deyim )
{
    deyimler;
}
```

Eğer *if* yapısı bir tek deyim içeriyorsa, { } blok parantezi kullanılmayabilir:

```
if ( mantıksal_deyim )
    deyim ;
```

Ama *if* yapısına bağlı olarak işlenmesi gereken birden çok deyim varsa, onların hepsini bir blok içine almak gerekir. Tek deyim olduğu için { } bloku kullanılmasa bile, biz ona *if bloku* diyeceğiz.

Eğer *mantıksal_deyim* (boolean) `true` değerini alıyorsa, *if bloku* içindeki *deyimler* sırasıyla işlenir; sonra program akışı *if* blokundan sonraki deyimi işlemeye başlar. Eğer *mantıksal_deyim* `false` değerini alıyorsa *if bloku* içindeki deyimler işlenmeden atlanır ve program akışı *if* blokundan sonraki deyime atlar.

*Örnekler:*

**Program 5.2.**

```
#include <stdio.h>
#include <locale.h>

int main()
{
  setlocale(LC_ALL,"");
  int n;

  printf("Yaşınız kaçtır? \n");
  scanf("%d"  , &n);

  if (n > 18 )
    printf("Siz bir seçmensiniz! \n");

  return 0;
}
```

```
/**
  Bu tek bir if yönlendirmesidir.
  x >= y olduğunda if blokuna girilmez.
*/
```

Program 5.3, kullanıcının girdiği iki sayıyı karşılaştırıyor. Birinci sayı küçükse, öyle olduğunu yazıyor. Değilse, başka bir iş yapmıyor. Karşılaştırma için ilişkisel operatörlerden küçüktür ( < ) operatörünü kullanıyor.

**Program 5.3.**

```c
#include <stdio.h>

int main()
{
  int x,y;
  printf("iki tamsayı giriniz. Sayılar arsında enaz bir boşluk
   bırakınız. \n");
  scanf("%d %d"  ,&x,&y);
  if(x < y)
    printf("%d sayisi %d den kucuktur\n" ,x,y);
}
```

```c
/**
  Bu tek bir if yönlendirmesidir.
  x >= y olduğunda if blokuna girilmez.
*/
```

## 5.4 Ardışık if deyimleri

`if` blokları ardışık olabilir. Bu durumda her bir `if` bloku, ötekilerden bağımsız çalışır. Birden çok ard arda if bloku olduğunda, mantıksal ifadeye bağlı olarak, program akışı hiç birine uğramayabilir, birisine ya da birden çoğuna uğrayabilir.

Örnek olarak, Program 5.3'yi biraz geliştirelim. Program 5.4, gene kullanıcının girdiği iki sayıyı karşılaştırıyor. Üç mümkün durumu tek tek denetliyor. Hangisi doğru ise onu yazıyor. Burada `if` deyimlerinin ard arda yazıldığına dikkat ediniz. Dolayısıyla program akışı herbiri için mantıksal deyimin doğru olup olmadığını denetliyor. Verilen iki sayı için, o üçünden yalnızca birisi doğru olabilir. Program akışı, yalnızca doğru olan if ... deyimine sapıyor; ötekileri geçiyor. Buradaki üç tane if deyiminin her birisi tek başına yalın bir if deyimidir; ötekilerden bağımsız çalışır. Karşılaştırma için ilişkisel operatörler dediğimiz $<$, $>$,$==$ operatörlerini kullanıyor.

**Program 5.4.**

```c
#include <stdio.h>
#include <locale.h>

int main()
{
  setlocale(LC_ALL,"");
  int x,y;
  printf("iki tamsayı giriniz. \n");
  printf("Girişte sayılar arasında enaz bir boşluk bırakınız. \n");
  printf("Ya da her sayıdan sonra Entere basınız \n");

  scanf("%d %d"  ,&x,&y);
  if(x < y)
```

```
    printf("%d sayisi %d den kucuktur\n" ,x,y);
  if(x > y)
    printf("%d sayisi %d den buyuktur\n" ,x,y);
  if(x == y)
    printf("%d sayisi %d ye esittir\n"  ,x,y);
  return 0;
}
```

```
/**
  Bu program ardışık if yönlendirmesine örnektir.
  3. ve 7. satırlar birlikte, gcc derleyicisine Türkçe karakterleri
   yazdırır.
*/
```

*Uyarı:*

Bir if yönlendirmesinde işlenen tek deyim ise, onu { } bloku içine almaya gerek yoktur. Alınsa da alınmasada da olur. Ama if yönlendirmesi birden çok deyimi içeriyorsa, o deyimlerin hepsi { } bloku içine alınır. Blok parantezleri kullanılsa da kullanılmasa da, bunlara *if bloku* diyoruz.

Program 5.1'de gün numarası 1 ile 7 arasında ise yalnızca ilgili if blokunu işler. Program akışı başka if blokuna uğramaz. Girilen gün numarası 1-7 arasında değilse, program akışı hiç bir if blokuna girmez.

**Örnek 5.1.**

```
#include <stdio.h>
#include <locale.h>

int main() {
  setlocale(LC_ALL,"");
  int gun;

  printf("Haftanın kaçıncı günündeyiz? \n");
  scanf("%d"  , &gun);

  if (gun == 1 )
    printf("Bu gün Pazartesi'dir' \n");

  if (gun == 2 )
    printf("Bu gün Salı'dır' \n");

  if (gun == 3 )
    printf("Bu gün Çarşamba'dır' \n");

  if (gun == 4 )
    printf("Bu gün Perşembe'dir' \n");

  if (gun == 5 )
    printf("Bu gün Cuma'dır' \n");

  if (gun == 6 )
    printf("Bu gün Cumartesi'dir' \n");
```

```
  if (gun == 7 )
    printf("Bu gün Pazar'dır' \n");

  return 0;
}
```

Buradaki gibi olası seçenekler çok olursa, *switch-case* yapısını kullanmak daha kolaydır (bkz. Kesim **??**).

## 5.5 if-else Yapısı

Bazı durumlarda önümüze iki seçenek çıkar. Belirli bir koşulun sağlanması durumunda seçeneklerden birinin, aksi halde ötekinin işlemesi istenebilir. Başka bir deyişle, bir koşulun sağlanıp sağlanmamasına bağlı olarak, iki seçenekten birisini mutlaka yaptırmak gerekir. Bu durumda, *if-else* yapısını kullanırız. Bu yapının sözdizimi şöyledir:

```
if (mantıksal-deyim)
    deyim-1;
else
    deyim-2;
```

Eğer deyim-1 ve deyim-2 yerinde işlenecek birden çok deyim varsa, onlar birer blok içine alınmalıdır:

```
if ( mantıksal-deyim )
  {
    deyimler-1
  }
else
  {
    deyimler-2
  }
```

Eğer mantıksal-deyim *true* değerini alıyorsa deyim(ler)-1 işlenir ve program akışı *if* denetim yapısından sonraki deyimi işlemeye başlar. Eğer mantıksal-deyim *false* değerini alıyorsa deyim(ler)-1 işlenmeden atlanır ve deyim(ler)-2 işlenir. Sonra, program akışı *if* denetim yapısından sonraki deyimi işlemeye başlar. Bu yapıda program ya deyim(ler)-1 ya da deyim(ler)-2 'yi işler. Bu yapı ardışık iki if... bloku ile yazılabilir görünüyor. Ama ardışık if bloklarının hiç birisi işlemeyebilir, bazısı ya da hepsi işleyebilir. *if-else* yapısında ise if ve else bloklarının ikisi birden işleyemez ama ikisinden birisini işlemeden geçemez.

Program 5.5, kullanıcının girdiği sayının pozitif mi, yoksa negatif mi olduğunu söylüyor. Program iki seçenekten birisini ve yalnızca birisini işler.

**Program 5.5.**

```
#include <stdio.h>
#include <locale.h>

int main()
{
  setlocale(LC_ALL , "");
  int a;
  printf("Bir tamsayı giriniz \n");
  scanf("%d", &a);

  if (a>0)
    printf("Girgiğiniz %d sayısı pozitiftir. \n", a);

  else
    printf("Girgiğiniz %d sayısı negatiftir. \n", a);

  return 0;
}
```

**Uyarı 5.2.**

Programda, girilecek a sayının tamsayı olması isteniyor. Ama kullanıcı, tamsayı yerine klavyeden *float, char* ya da *string* girebilir. C dili kullanıcının girdiği verinin tipinin doğru olup olmadığını denetlemez. Girilen veriyi *int* olarak okur. Örneğin, Program 5.5, tamsayı girilmesini istediği zaman kullanıcı;

- 654.321 float sayısını girmiş olsun.

  ```
  Bir tamsayı giriniz
  654.321
  Girdiğiniz 654 sayısı pozitiftir.
  ```

  olur. Görüldüğü gibi, derleyici girilen sayının tam kısmını alıyor. Kesir kısmını atıyor. Aldığı sayıyı a değişkeninin değeri olarak işlemde kullanıyor.

- Önce M tuşuna arkasından Enter tuşuna basılmış olsun. Ama derleyici o anda &a adresinde yazılı olan binary veriyi int olarak okur ve okuduğu o sayıya göre işlem yapar.

- Son olarak a tamsayısı istenince bir string girelim. Gene derleyici aynı işi yapar. &a adreinde yazılı olan binary veriyi int olarak okur ve okuduğu o sayıya göre işlem yapar.

Buradan çıkaracağımız sonuç şudur:

**Uyarı 5.3.** *C dili veri tipini denetlemez.*

Değişkene, bildirimdekinden farklı bir veri tipi atanabilir. Derleyici, o anda adresteki binary değeri istenen tipmiş gibi okur. Tip denetimi yapmaması derleyiciye hız kazandırır; ama kullanıcı dikkatli olmazsa koşma hataları (run time error) oluşabilir. Bilindiği gibi, koşma anı hataları, bilgisayar programcılığında ölümcül hatalardan sayılır. Çünkü programın hatalı iş yaptığı bilinmeyebilir.

**Uyarı 5.4.** if ... denetimi *için mutlaka bir* mantıksal_deyim *(boolean) gereklidir.*

Program 5.6, kullanıcının girdiği sayının çift mi, yoksa tek mi olduğunu söylüyor. Program iki seçenekten birisini ve yalnızca birisini işler.

**Program 5.6.**

```
#include <stdio.h>
#include <locale.h>

int main()
{
  setlocale(LC_ALL , "");
  int a;
  printf("Bir tamsayı giriniz \n");
  scanf("%d", &a);


  if (a%2)
    printf("Girgiğiniz %d sayısı çifttir \n", a);

  else
    printf("Girgiğiniz %d sayısı tektir \n", a);

  return 0;
}
```

Program 5.7 verilen bir yılın artık yıl olup olmadığını bulur.

**Program 5.7.**

```
#include <stdio.h>
#include <locale.h>

int main() {
  setlocale(LC_ALL , "");

  int sene;

  printf("Hangi yıl? (yyyy) : \n");
  scanf("%d", &sene);

  if ((sene % 4 == 0 && sene % 100 != 0) || sene % 400 == 0)
    printf(" %d yılı artık yıldır. Şubat ayı 29 çeker. \n", sene);
```

```
  else
    printf(" %d yılı artık değildir. Şubat ayı 28 çeker. \n",  sene);

  return 0;
}
```

```
/**
 Hangi yıl? (yyyy) :
 1997
 1997 yılı artık yıl değildir. Şubat ayı 28 çeker.
*/
```

**Uyarı 5.5.**

Ardışık if ... yapısına benzer olarak, ardışık *if-else* yapıları kurulabilir. Bu yapıda her bir *if-else* bloku ötekilerden bağımsız çalışır. Ama program saçaklanır, doğrusal akmaz.

### 5.5.1 Çoklu Saçaklar

Bazen program akışının yönlenebileceği seçenekler *çoklu saçaklara* ayrılabilir. Bu durumlarda *if-else* ya da *else-if* bloklarından oluşan çoklu durum yapıları oluşur. İlk *if-else* yapısında ikiye ayrılan seçenekler, her adımda tekrar ya `if` ya da `else` tarafında ikiye ayrılarak bir saçak oluştururlar. Sözdizimi şöyledir:

```
  if (boolean-1)
    {
    Deyimler-1;
    }
  else if (boolean-2)
    {
    Deyimler-2;
    }
  else if (boolean-3)
    {
    Deyimler-3;
    }
  ...
  else
    {
    Deyimler-n;
    }
```

```
/**
Bu yapıda gerektiği kadar else-if deyimi kullanılabilir. Her adımda
 program akışı ya if tarafına ya da else tarafına yönlenir.
 Adımlardan hiç birisi koşulu (boolean) sağlamıyorsa, son deyim
 olarak else bloku işler.
*/
```

### Örnekler

Program 5.8 bir üçgenin üç kenarını okuyor. Sonra *"iki kenar toplamı üçüncü kenardan büyüktür"* kuralını uygulayarak, girilen kenar uzunluklarının bir üçgen oluşturup oluşturmadığını denetliyor. Denetleme işini main() dışında tanımlanan ucgen() fonksiyonu yapıyor. main(), o fonksiyonu gerçek parametre değerleriyle çağırıyor. Tabii, istenirse, ucgen() fonksiyonunun gövdesi main() 'nin gövdesi içine yazılabilir. Ama büyük programlarda, her farklı işi bir fonksiyona yaptırmak daha iyidir.

**Program 5.8.**

```
#include <stdio.h>
#include <locale.h>

main() {
  setlocale(LC_ALL , "");
  int a , b , c ;
  printf("Üçgenin kenar uzunluklarını giriniz : a,b,c = ? ");
  scanf("%d %d %d",&a,&b,&c);
  ucgen(a,b,c);
}

ucgen(x,y,z)
int x,y,z;
{
  if ( x+y <= z) {
    printf("z kenarı çok büyük. Bu bir üçgen değildir.\n");
  } else  if( x+z < y) {
    printf("y çok büyük. Bu bir üçgen değildir.\n");
  } else if(y+z < x) {
    printf("x çok büyük. Bu bir üçgen değildir.\n");
  } else
    printf("Bu bir ucgendir.\n");
}
```

Program 5.9, kullanıcının girdiği ısı derecesine göre, günün sıcak olup olmadığını söylüyor.

**Program 5.9.**

```
#include <stdio.h>
#include <locale.h>

main()
{
  setlocale(LC_ALL , "");

  int i;

  printf("Bugün ısı kaç derecedir? ");
  scanf("%d",  &i);
```

```
  if (i < 10)
    printf("Bugün hava soğuktur. \n");
  else if (i < 20)
    printf("Bugün hava serindir. \n");
  else
    printf("Bugün hava sıcaktır. \n");
}
```

```
/**
  Bugün ısı kaç derecedir?
  -5
  Bugün hava soğuktur.
*/
```

Program 5.10, kullanıcının girdiği sayının 3 ile bölünebildiğini, çift olduğunu, tek olduğunu ya da tek olduğu halde 3 ile bölünemediğini söylüyor.

**Program 5.10.**

```
#include <stdio.h>
#include <locale.h>

main()
{
  setlocale(LC_ALL , "");

  int i;

  printf("Bir tamsayı giriniz \n");
  scanf("%d", &i);

  if (i % 3 == 0)
    printf("%d tamsayısı 3 ile tam bölünür. ", i);
  else if (i % 2 == 0)
    printf("%d tamsayısı çitf sayıdır. ", i);
  else
    printf("%d tamsayısı tek sayıdır, ama 3 ile tam bölünmez. ", i);
}
```

```
/**
  Bir tamsayı giriniz
  27
  27 tamsayısı 3 ile tam bölünür.
*/
```

*if-else* ve *else-if* yapıları istenildiği kadar içiçe konulabilir. Program 5.11, kullanıcının girdiği puana göre karne notunu söylüyor.

**Program 5.11.**

```
#include <stdio.h>
#include <locale.h>
```

```
main()
{
  setlocale(LC_ALL , "");

  int i;

  printf("Not ortalamanızı giriniz : \n");
  scanf("%d", &i);

  if (i > 90)
    printf("Puanınız %d ise notunuz A olur. \n", i);
  else if (i > 75)
    printf("Puanınız %d ise notunuz B olur. \n", i);
  else if (i > 60)
    printf("Puanınız %d ise notunuz C olur. \n", i);
  else if (i > 50)
    printf("Puanınız %d ise notunuz D olur. \n", i);
  else
    printf("Puanınız %d ise notunuz F olur. \n", i);

  return 0;
}
```

```
/**
  Not ortalamanızı giriniz :
  92
  Puanınız 92 ise notunuz A olur.
*/
```

Program 5.12, kullanıcının girdiği karakterin tipini söylüyor.

**Program 5.12.**

```
#include <stdio.h>
#include <locale.h>

main()
{
  setlocale(LC_ALL , "");

  char ch;

  printf("Bir tuşa basınız :");
  scanf("%c", &ch);
  if (isupper(ch))
    printf("Büyük harf girdiniz.");
  else
  if (islower(ch))
    printf("Küçük harf girdiniz.");
  else
  if (isdigit(ch))
    printf("Bir rakam girdiniz.");
  else
    printf("Alfasayısal olmayan bir karekter girdiniz.");
}
```

```
/**
  Programın farklı koşmalarında, girilen karektere bağlı olarak
   çıktılar şuna benzer.
  Bir tuşa basınız: H
  Büyük harf girdiniz.

  11.satır yerine ch = getchar(); yazılabilir.
*/
```

**Uyarı 5.6.** *karakter ve string fonksiyonları için bkz. Bölüm* **??**.

Program 5.13, [1,100] aralığında 10 tane rasgele sayı üretiyor. her birisinin tek mi yoksa çift mi olduğunu buluyor.

**Program 5.13.**

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

int main()
{
    setlocale(LC_ALL , "");
  int c, r;
  for (c = 1; c <= 10; c++)
  {
    r = rand() % 100 + 1;
    printf("%d", r);

  if (r % 2)
  {
    printf("  Sayı çifttir. \n");
  }
  else
  {
    printf("  Sayı tektir. \n");
  }
    }
}
```

```
/**
42  Sayı tektir.
68  Sayı tektir.
35  Sayı çifttir.
1  Sayı çifttir.
70  Sayı tektir.
25  Sayı çifttir.
79  Sayı çifttir.
59  Sayı çifttir.
63  Sayı çifttir.
65  Sayı çifttir.
*/
```

### `switch-case` Yapısı

Program akışının yönlenebileceği olası seçenekler çoksa, *ardışık if*, ya da *saçak yapıları* yerine *switch* yapısı daha kolay olur.

```
switch (seçici)
{
   case seçki-1:
      deyim-1;
      break;

   case seçki-2:
      deyim-2;
      break
      ...

   [default:
      deyim
      break]
}
```

Bu yapıda *case* ifadelerinin sayısı için bir kısıt yoktur, gerektiği kadar *case* bloku konulabilir.

Bu sözdiziminde geçen terimlerin açıklanması:

**switch(seçici):** Seçici bir değişken ya da bir ifade olabilir. Seçici değişken olduğunda *int* ya da *char* tiplerinden herhangi birisinden bir literal (sabit değer) almalıdır. Seçici bir deyim ise bu türlerden literal bir değer veren formüldür. Burada formül, matematiksel formüllerde olduğu gibi, bir sonuç veren deyim olarak algılanmalıdır.

**case seçki:** Seçici değişkenin ya da seçici ifadenin aldığı değerdir. O değeri, çoklu yol ayrımında program akışına yol gösteren bir bir levha, bir etiket gibi düşünebiliriz. Bu değer seçicinin alabildiği yukarıdaki veri tiplerinden herhangi birisi olabilir. Tabii, bu değer bir literal (sabit değer) olmalıdır. *int* ya da *char* değerlerin artan sırada dizilmesi önerilir; ama zorunlu değildir.

**case_bloku:** Bir case seçeneğinde işlenecek deyimlerden oluşan bloktur. Blokta birden çok deyim olabilir. Her case_blokundaki deyimler belirli olduğundan { } blok parantezine gerek yoktur.

**break:** Bir case seçeneğine giren program akışı, o seçenekteki deyimleri işleyince, sonrakilere geçmeden switch yapısının dışına çıkmalıdır. Bu çıkışı *break* anahtar sözcüğü sağlar. *break;* deyimi kullanılmazsa, program akışı, sonraki case seçeneklerinin hepsini çalıştırır. *break;* deyimi, program akışını *switch-case* yapısından çıkarır ve switch-case yapısından sonraki ilk deyime gönderir.

**default:** İsteğe bağlı (optional) bir seçenektir. Seçicinin değeri, *case* ile belirlenen hiçbir seçki ile uyuşmadığı zaman yapılmasını istediğimiz işler için gerekli deyim(ler)i bu seçeneğe yazarız. *case* seçeneklerinden hiçbiri gerçekleşemezse *default* seçeneği çalışır. Seçicinin değerinin, case'lerdeki seçkilerden birisine eşit olacağı kesinlikle biliniyorsa, *default* seçeneğini yazmanın gereği yoktur.

*switch* yapısının nasıl kullanıldığını aşağıdaki örnekte göreceğiz.

Program 5.14, *switch-case* yapısında seçici olarak *int* tipi değişken kullanıyor. `case` bloklarının sonunda *break;* deyimi olmadığı için, ilk seçkiden sonra gelen `case` bloklarının hepsi çalışır. Ama böyle olması genellikle istenilen iş değildir.

**Program 5.14.**

```
#include <stdio.h>
#include <locale.h>

main() {
  setlocale(LC_ALL , "");
  int x;
  scanf("%d", &x);
  switch(x) {
    case 1:
      printf(" 1. konum.\n");
      printf(" Birinci adım\n");
    case 2:
      printf(" 2. konum.\n");
      printf(" ikinci adim\n");
    case 3:
      printf(" 3. konum.\n");
      printf(" Üçüncü adım\n");
    case 4:
      printf(" 4. konum.\n");
      printf(" Dördüncü adım\n");
    default :
      printf("Hiç adım tılmadı\n");
  }
}
```

```
/**
  case seçeneklerinde  break; deyimi olmadığı icin seçilen case
   blokundan  sonraki  bütün case blokları çalışır.

  seçki \emph{int} tipten ise, değeri  scanf() fonksiyonu ile
   okunabilir.
*/
```

Program 5.15, *switch-case* yapısında seçici olarak *char* tipi değişken kullanıyor. case seçeneklerinin sonunda *break;* deyimi olduğu için, program

akışı bir case blokunu işleyince sonrakilere uğramadan switch-case yapısı dışına çıkar. Yapıdan sonraki ilk deyime gider.

**Program 5.15.**

```
#include <stdio.h>
#include <locale.h>

main() {
  setlocale(LC_ALL , "");
  int ch;
  ch = getchar();
  switch(ch) {
    case 'M':
    case 'm':
      printf(" 1. konum.\n");
      printf(" Birinci adım\n");
      break;
    case 'P':
    case 'p':
      printf(" 2. konum.\n");
      printf(" ikinci adim\n");
      break;
    case 'R':
    case 'r':
      printf(" 3. konum.\n");
      printf(" Üçüncü adım\n");
      break;

    case 'B':
    case 'b':
      printf(" 4. konum.\n");
      printf(" Dördüncü adım\n");
      break;
    default :
      printf("Hiç adım atılmadı\n");
  }
}
```

```
/**
  case seçeneklerine  break deyimi konulduğu için, seçkiye bağlı
   olarak işlenen case blokundan sonra program akışı switch-case
   yapısı dışına çıkar;  yapıdan sonraki deyime geçer.

  char tipinden olan  seçki değerini elde etmek için  getchar()
   fonksiyonu kullanılıyor.

  Seçki olarak, kullanıcının büyük ya da küçük harf girebilmesi için,
    her iki seçenek ayrı ayrı yazıldı.
*/
```

Program 5.2, switch-case yapısında seçici olarak int tipi değişken kullanılmasına örnektir.

**Örnek 5.2.**

```c
#include <stdio.h>
#include <locale.h>

main() {
  setlocale(LC_ALL , "");
  int ay;

  printf("Kaçıncı ay? \n");
  scanf("%d", &ay);


  switch (ay) {
    case 1:
      printf("Ocak");
      break;
    case 2:
      printf("Şubat");
      break;
    case 3:
      printf("Mart");
      break;
    case 4:
      printf("Nisan");
      break;
    case 5:
      printf("Mayıs");
      break;
    case 6:
      printf("Haziran");
      break;
    case 7:
      printf("Temmuz");
      break;
    case 8:
      printf("Ağustos");
      break;
    case 9:
      printf("Eylül");
      break;
    case 10:
      printf("Ekim");
      break;
    case 11:
      printf("Kasım");
      break;
    case 12:
      printf("Aralık");
      break;
  }
}
```

```
/**
  Kaçıncı ay
  7
  Temmuz
*/
```

Program 5.16 *switch-case* yapısında *seçici* olarak, *char* tipten literal değer alan bir deyim kullanılmasına örnektir. Kullanıcının girdiği harfi *seçki deyimi* (toupper(ch)) büyük harfe çevirmektedir.

**Program 5.16.**

```
#include <stdio.h>
#include <locale.h>

main() {
  setlocale(LC_ALL , "");

    char ch;

  printf("Karne Notunuzu giriniz \n");
  ch = getchar();


    switch (toupper(ch)) {
    case 'A':
      printf("Pekiyi");
      break;

    case 'B':
      printf("İyi");
      break;

    case 'C':
      printf("Orta");
      break;

    case 'D':
      printf("Hmmmm....");
      break;

    case 'F':
      printf("Daha iyisini başarabilirsin!");
      break;

    default:
      printf("Başarı notunu yanlış girdiniz!");
      break;
    }
  }
```

```
/**
  Karne notunuzu giriniz :
  a
  Pekiyi
*/
```

Program 5.17, seçici değişken yerine bir deyimin kullanılabileceğini gösteriyor. *(puan / 10)* deyimi kullanıcının 100 tam puan üzerinden girdiği puanı 10 tam not üzerinden değerlendirme sistemine dönüştürmektedir.

**Program 5.17.**

```
#include <stdio.h>
#include <locale.h>

main() {
  setlocale(LC_ALL , "");

  int puan;

  printf("Puanınızı giriniz? \n");
  scanf("%d", &puan);

  switch (puan / 10) {
    case 10:
      printf("Notunuz A dır. ");
      break;
    case 9:
      printf("Notunuz A- dır. ");
      break;
    case 8:
      printf("Notunuz B+ dir. ");
      break;
    case 7:
      printf("Notunuz B dir. ");
      break;
    case 6:
      printf("Notunuz B- dir. ");
      break;
    case 5:
      printf("Notunuz C+ dir. ");
      break;
    case 4:
      printf("Notunuz C dir. ");
      break;
    case 3:
      printf("Notunuz C- dir. ");
      break;
    case 2:
      printf("Notunuz D+ dir. ");
      break;
    case 1:
      printf("Notunuz D dir. ");
      break;
    case 0:
      printf("Notunuz F dir. ");
      break;
    default:
      printf("Yanlış not girdiniz! ");
      break;
  }
}
```

```
/**
Not ortalamanız nedir?
100
Notunuz A dır.
*/
```

### 5.5.2 Koşullu Operatör

`?:` operatorü

Bir değişkenin değeri bir mantıksal ifadenin doğru ya da yanlış olmasına bağlı olarak belirlenebiliyorsa, onu *if-else* deyimi ile tanımlamak yerine, daha kısa olarak *koşullu deyim* (üçleme, conditional operator, ternary conditional) ile tanımlayabiliriz. Örneğin, a ile b sayılarının maksimumunu if-else yapısı ile

```
if (a > b) {
  max = a;
}
else {
  max = b;
}
```

biçiminde bulabiliriz. Koşullu deyim kullanarak, bunu daha kısa olarak,

```
max = (a > b) ? a : b;
```

biçiminde yazabiliriz. Bu deyimin sonucu, eğer $(a > b)$ ise $a$, değilse $b$ dir.

Genel olarak,

**Tanım 5.1.**

```
e1 ? e2 : e3
```

deyimi, bir operatördür, dolayısıyla bir değer verecektir. $e1$ bir mantıksal deyimdir (boolean), $e2$ ve $e3$ literal değerlerdir. Eğer $e1$ *true* ise, 5.1 operatörünün değeri $e2$, *false* ise $e3$ olur.

Bu operatöre koşullu (üçleme, conditional, ternary) operatör denilir.

En genel olarak , koşullu operatörün sözdizimini (sytax)

**Tanım 5.2.**

```
mantıksal−deyim ? terim1 : terim2;
```

biçimindedir. Burada *mantıksal-deyim* `true` ya da `false` değerini alan bir boolean deyimdir. *terim1* ve *terim2* literal (sabit) değerlerdir ya da literal değerler veren iki deyimdir. Örneğin,

```
int c = (1 > 2) ? (c = 25) : (c =18);
```

operatörünün değeri 18 olur.

Program 5.18, verilen iki sayının minimumunu buluyor.

**Program 5.18.**

```c
#include <stdio.h>
#include <locale.h>

int main()
{
  setlocale(LC_ALL , "");
  int a, b, min ;

  printf("İki tamsayı giriniz \n");
  scanf("%d %d", &a, &b);

  printf("Girgiğiniz sayılar %d ve %d dir.r \n", a, b);

  min = ((a < b)? a: b);

  printf("Küçük olan sayı : %d dir\n", min);

  return 0;
}
```

```
/**
  İki tamsayı giriniz
  23 67
  Küçük olan sayı : 23 dir.
*/
```

Program 5.19, koşullu operatörün mantıksal ifadelere de uygulanabileceğini gösteriyor.

**Program 5.19.**

```c
#include <stdio.h>
#include <locale.h>

int main()
{
  setlocale(LC_ALL , "");
  int a=5, b, min ;

    b = (3 > 2) ? (3 > a) : (3 < a);
    printf("%d\n", (3 > a));
    printf("%d\n", b);

    if (3 > 2)
      b = (3 > a);
    else
      b = (3 < a);
    printf("%d\n", b);
  }
```

```
/**
  0
  0
  0
*/
```

## 5.6 Alıştırmalar

1. Program 5.3 else-if yapısına dönüştürünüz.

   **Örnek 5.3.**

```
#include <stdio.h>
#include <locale.h>

int main() {
  setlocale(LC_ALL, "");
  int gun;

  printf("Haftanın kaçıncı günündeyiz? \n");
  scanf("%d"  , &gun);
  switch (gun) {

    case 1:

      printf("Bu gün Pazartesi'dir' \n");
      break;

    case 2:
      printf("Bu gün Salı'dır' \n");
      break;

    case 3:
      printf("Bu gün Çarşamba'dır' \n");
      break;

    case 4:
      printf("Bu gün Perşembe'dir' \n");
      break;

    case 5:
      printf("Bu gün Cuma'dır' \n");
      break;

    case 6:
      printf("Bu gün Cumartesi'dir' \n");
      break;

    case 7:
      printf("Bu gün Pazar'dır' \n");
      break;

    default:
      printf("Haftada öyle bir gün yok! \n");

      return 0;
  }
}
```

2. Program 5.20'i çözümleyiniz.

   **Program 5.20.**

```
#include <stdio.h>
#include <locale.h>

int g;
void  tatil_mi(g) {
  int gun = g;
  if (gun> 0 & gun< 6)
    printf("Bu gün iş günüdür. Çalışmalısın \n");
          if (gun == 6 || gun == 7)
          printf("Bu gün hafta sonudur. Dinlenmelisin \n");
        }

    int main()
       {
       setlocale(LC_ALL,"");
       int n;

       printf("Bu gün haftanın kaçıncı günüdür? \n");
       scanf("%d"  , &n);

       tatil_mi(n);

       return 0;
          }
```

3. Yalın if-else yapısında kaç seçenek mümkündür?

4. 
```
int c = (7 > 9) ? 25 : 45;
```

   operatörünün sonucu nedir?

5. Yukarıdaki koşullu operatörü *if-else* yapısı haline getiriniz.