Kontrol yapıları ve Döngüler

Ders notları
(1) C How to Program 8th Edition, by Deitel & Deitel
(2) Bilgisayar Programlama III notları – (Dr. Süha Tuna)

İÇİNDEKİLER

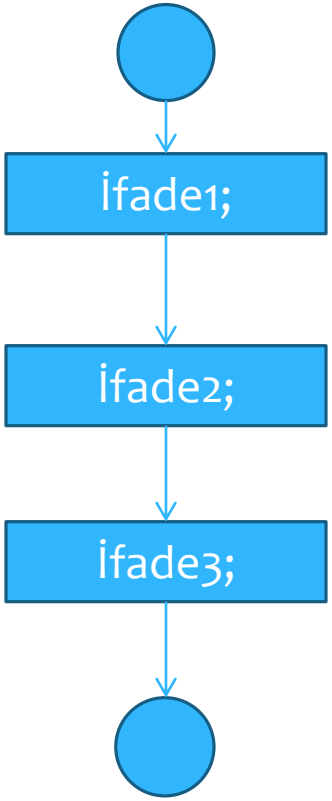
* Kontrol yapıları
  * `if`
  * `switch`
* Döngü yapıları
  * `for`
  * `while`
  * `return`
  * `goto`

Kontrol Yapıları

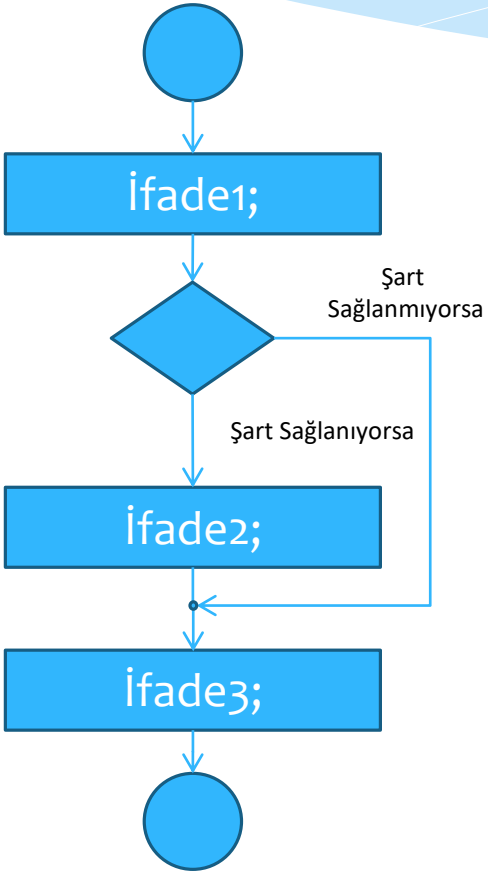
* Normalde kod akışı sıralıdır ve baştan sona satır satır çalıştırılır.
* Ancak yazılımlarda da gerçek hayatın yansıması olarak çeşitli şartlara göre yönlendirmeler ve tekrarlar gerekir.
* Kontrol ve döngü yapıları kodun istenilen şartlara göre yönlendirilmesini sağlar.

Kontrol ve Döngü Yapıları

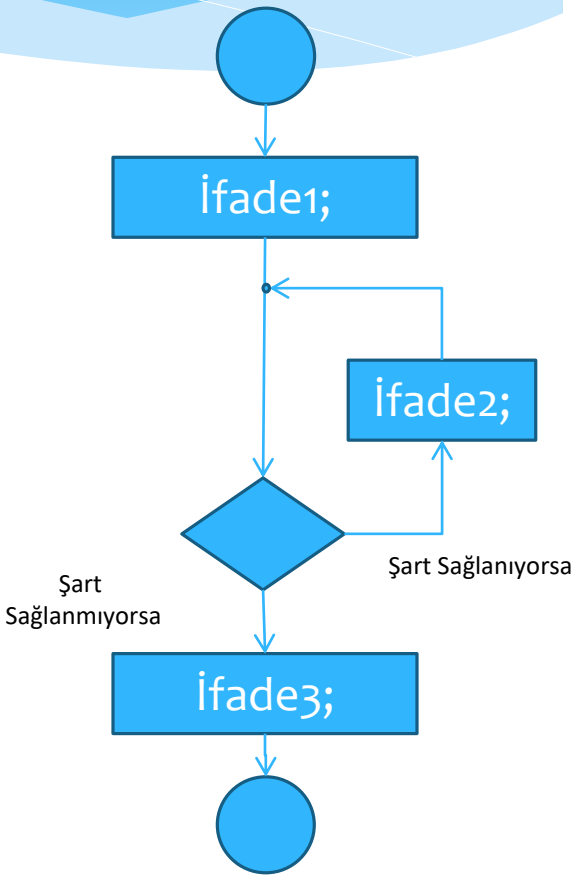
Normal kod akışı



Kontrol yapıları kullanıldığında kod akışı



Döngü Yapıları



if Bloğu

* `if`, kodun bir kısmının bir şarta bağlı çalışması gerektiği zaman kullanılır.
* Yazılışı

```
if (şart) {
    Şarta Bağlı İşlem
}
```

* Şart `true` dönerse `if` bloğu içindeki işlem çalıştırılır.
* Şart `false` dönerse `if` bloğu içindeki işlemler atlanır ve kod `if` bloğu sonundan çalıştırılmaya devam edilir.

Eşitlik (==) ve Atama (=) Operatorleri ilgili sorunlar

* Bu tür sorunlar tehlikelidir.
  * Yazım hatası (Syntax errors) oluşturmaz.
  * Derlerken fark edilmez.
  * Değer üreten tüm eşitlikler Kontrol yapılarında kullanılabilir.
  * Sıfır haricindekiler `true`, sıfır değerler `false` kabul edilir.
  * Örnek ==:

```
if ( payCode == 4 )
   printf( "You get a bonus!\n" );
```

  * `payCode` eğer 4 ise `You get a bonus` yazılır.

Mantıksal (==) ve atama(=) Operatörlerinin net ayrılması

* Eğer == yerine = kullanılırsa:

```
if ( payCode = 4 )
    printf( "You get a bonus!\n" );
```

  * payCode 4 olarak değerlenir.
  * 4 sıfır değildir, ifade true olur
  * payCode ne olursa olsun if içerisine girilir.
* Mantıksal hata, yazım hatası değil

Else if

* Birden fazla şartın birlikte denenmesi için **`else if`** ifadesi kullanılır.

```
if
   İlk işlem
else if
   İkinci işlem
```

* Şart ilk doğru olan `if` işletilir.
* Örnek: Ayların numarasından ayların ilk harfini bulan bir algoritma tasarlayınız.

else

* Diğer tüm koşulların sağlanmadığı durumda çalışacak işlemler için **`else`** ifadesi kullanılır.

```
if
   İlk işlem
else if
   İkinci işlem
Else
   Diğer tüm koşulların sağlanmadığı durumda çalışacak
   işlemler
```

* Örnek: Bir parametre değeri 1 ile 10 arasındaysa «küçük», 10 ile 20 arasındaysa «büyük» diğer durumlarda «tanımsız yazan bir uygulama geliştiriniz.»

if örnek

* Örnek:

```
int a,b
…
if (a>b)
  a=b
else
  a=0
```

* if ve else'den sonra birden fazla işlem varsa {} kullanılır

switch

* Bir değişkenin değerlerine göre işletilecek kodun belirlenmesini sağlar.

```
switch(değişken){
   case değer1:
      İşlem1
      break;
   case değer2:
      İşlem2
      break;
      …
   case değerN:
      İşlemN
      break;
      }
```

* break case'den çıkmak için kullanılır. Kullanılmazsa?
* Örnek: Numarasında ayların ilk harfini bulan bir fonksiyon tasarlayınız.
* Birden beşe kadar girilen bir numaranın değerini ekrana yazdıran bir fonksiyon tasarlayınız.

Switch Örnek: Ayların Numarasından İsmini bulma

```
void main() {
        int month = 8;
        char monthString[10];
        switch (month) {
            case 1:  monthString = "January";                  break;
            case 2:  monthString = "February"                  break;
            case 3:  monthString = "March";                    break;
            case 4:  monthString = "April";                    break;
            case 5:  monthString = "May";                      break;
            case 6:  monthString = "June";                     break;
            case 7:  monthString = "July";                     break;
            case 8:  monthString = "August";                   break;
            case 9:  monthString = "September";                break;
            case 10: monthString = "October";                  break;
            case 11: monthString = "November";                 break;
            case 12: monthString = "December";                 break;
            default: monthString = "Invalid month";            break;
        }
        printf("%s", monthString);
    }
```

switch Örneğinin if ile yapılması:

Ayların Numarasından İsmini bulma

```
int month = 8;
if (month == 1) {
    printf("January");
} else if (month == 2) {
    printf("February");
}
...  // devam
```

switch yapısında break unutulmamalıdır!

```
void main() {
       char futureMonths  [15] ;
       int month = 8;
       switch (month) {
           case 1:  futureMonths="January";
           case 2:  futureMonths="February";
           case 3:  futureMonths="March";
           case 4:  futureMonths="April";
           case 5:  futureMonths="May";
           case 6:  futureMonths="June";
           case 7:  futureMonths="July";
           case 8:  futureMonths="August";
           case 9:  futureMonths="September";
           case 10: futureMonths="October";
           case 11: futureMonths="November";
           case 12: futureMonths="December";
                    break;
           default: break;
       }
       if (! futureMonths) {
           printf("Invalid month number");
       }
       else{
           printf("Choseen month is %s", futureMonths);
       }
   }
```

Kod çıktısı aşağıdaki şekilde olur.

December

Oysa beklenen sadece «August» gelmesidir.

Break kullanılmadığından ilk uyan «case değer» kısmından sonraki tüm kısımlar çalıştırılır.

Özel durumlar dışında her case'de break olmalıdır!

Her case'e break eklenmeyen bir özel durum örneği(Birden fazla label için aynı işlemin yapılması):

Ayların gün sayılarını gösteriniz.

```
void main() {

    int month = 2;
    int year = 2000;
    int numDays = 0;

    switch (month) {
        case 1: case 3: case 5: case 7: case 8: case 10:
        case 12:
            numDays = 31;
            break;
        case 4: case 6:  case 9: case 11:
            numDays = 30;
            break;
        case 2:
            if (((year % 4 == 0) &&  !(year % 100 == 0)) || (year % 400 == 0))
                numDays = 29;
            else
                numDays = 28;
            break;
        default:
            printf("Invalid month.");
            break;
    }
    printf("Number of Days = %d ",  numDays);
}
```

`if` ve `switch` farkları

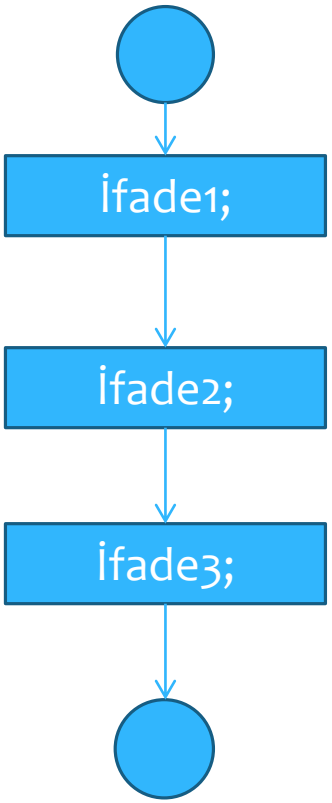
* `switch` sadece bir değişkenin değerine göre kontrol yapar.
* `switch` sadece eşitliği kontrol eder `if` ise her türlü mantıksal şartı çalıştırır.
* `switch` aynı değişkeni kontrol eden bir çok `if` yerine uygundur.
* `if`'in `switch` yapısına göre daha esnek olduğu söylenebilir.

Kontrol Yapıları

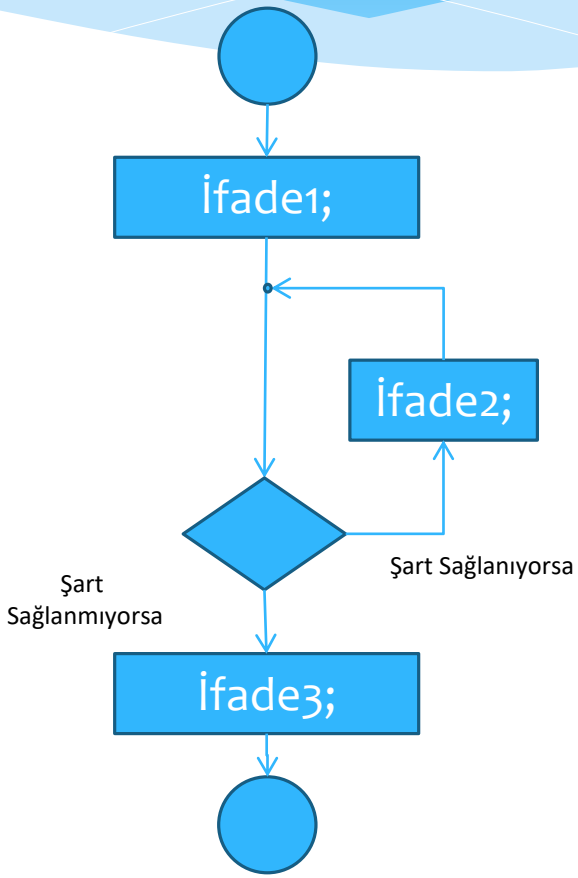
* Döngü yapıları belli kod parçalarının tekrarlı şekilde çalıştırılması için kullanılır.

Kontrol ve Döngü Yapıları

Normal kod akışı



Döngü Yapıları



Döngü ifadeleri

* while
* Do-while
* for

while

```
while (şart_ifadesi) {
  //çalıştırılacak kodlar
}
```

* Şart_ifadesi true olduğu sürece döngü içindeki kodları çalıştırmaya devam eder.
* Örnek: 1'den 10'a kadar olan numaraları ekrana yazdıran bir algoritma yazınız. Yazdırma işlemi nesne içerisinde olacaktır.
* Örnek: 1'den diğer bir sayıya kadar olan doğal sayıları toplayan bir algoritma yazınız.

while Örnek:

Birden 10'a kadar sayıların toplamını bulunuz.

```
int main(){
      int count = 1;
      while (count < 11) {
          printf("Toplam: %d " , count);
          count++;
      }
}
```

Do while

```
Do {
    //çalıştırılacak kodlar
}
While (şart)
```

* Şart `true` olduğu sürece döngü içindeki kodları çalıştırmaya devam eder.
* Döngü en az bir defa çalışacaktır.
* Örnek: 1'den diğer bir sayıya kadar olan numaraları toplayan bir nesne yazınız.

Do While Örnek: Birden 10'a kadar sayıların toplamını bulunuz.

```
int main(){
        int count = 1;
        do {
            printf("Count is: %d" , count);
            count++;
        } while (count < 11);
 }
```

for

* for (başlangıç_şartı; bitiş_şartı; artış)
  { işlemler }
* Örnek: for (i=1, i<10; i++) {…}

* Bitiş şartı false olduğu sürece döngü indeksi artar/azalır ve döngü içindeki kodları çalıştırmaya devam eder.
* for döngü indisi dışarıda tanımlanmalıdır.
* Birden fazla başlangıç şartı varsa araya virgül konarak yazılır.
  * for (i=1, j=10; i<j; i++,j--) {…}

* Örnek Çalışma: Bir sayının asal olup olmadığını bulan bir nesne tasarlayınız.

Örnek

Birden 10'a kadar sayıların toplamını bulunuz

```
void main(){
      int toplam=0;
      int i;
      for(i=1; i<11; i++){
            toplam=toplam+i;
      }
      printf(" Toplam: %d"  , toplam);
}
```

for ile sınırsız döngü oluşturma

* For ifadesinin tüm kısımları seçime bağlıdır.

* // Sonsuz döngü
* for (; ;) {
*
* // çalışacak kodlar.
* }

Kod içinde bir noktadan diğerine atlama ifadeleri

* **break**
* **continue**
* **goto**
* **return**

break

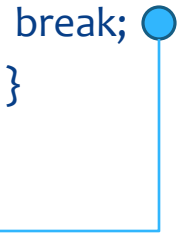
* **Döngüyü sonlandırmak için kullanılır.**
* **İçinde bulunduğu döngüyü sonlandırır. Kod döngü sonrası satırdan devam eder.**

Örnek: indeks 5 olursa döngüyü sonlandıracak bir uygulama tasarlayınız.

```
for (i=1, i<10; i++)
{…
    if (i==5) break;
    printf("%d",i);
}
```

break

```
For(){
    While (){
        For {
            İf (şart)
                {
                 break;
                }
            }
        ..
        }
    }
```



Break Örneği: Verilen aralıkta bir değerin olup olmadığını kontrol eden bir uygulama yazınız.

```
void main() {

    int altSinir=5,ustSinir=100;
    int searchfor = 12;
    int bulunduMu=0;
    int i=0;

    while (i = <ustSinir) {
       if (i == searchfor) {
          bulunduMu=1;
          break;
       }
              i++;
    }

    if (bulunduMu) {
       printf("Değer bulundu");
    } else {
       printf(" değer bulunamadı");
    }
}
```

continue

* Döngüdeki bir tekrarı (iteration) sonlandırmak için kullanılır.
* Örnek:

```
for (i=1, i<10; i++)
{…
    if (i==5) continue;
    printf("i: %d \n", i);
}
```

continue Örneği: Bir aralıktaki 7'ye bölünmeyen rakamlarının adedini bulunuz.

```
void main() {
      int numPs = 0;
      int i;
     for (i = altSinir; i < ustSinir; i++) {
         if (i%7== 0){
            continue;
         }
         numPs++;
     }
     printf("Bulunan rakam sayısı: %d  " , numPs );
  }
```

return

* Metoddan* çıkmak için kullanılır. Metodun dönüş türüne uygun bir return ifadesi kullanılmalıdır.

```
int getdeger()
{...
    printf("i %d \n",1);
    return;
    printf("i %d \n",2);
}
```

goto

Programın herhangi bir yerinden başka bir yerine atlamak için goto deyimi kullanılır. Goto komutunu kullanmak için , " Goto " isminin yanına , atlanmak istenen yerin sembolik ismi yazılır. .Goto ile bir döngünün içine atlanmasına izin yoktur , fakat bir döngünün dışına atlanabilir. Ayrıca bir fonksiyondan ötekine de "goto" ile geçilemez.

```
#include "stdio.h"
main( )
{
   int d1,d2;
  d2=0;
  etiket1:
  for (d1=1; d1<=5; d1++)
  printf("%d",d1);
  printf("\n");
  d2++;
  if (d2<3 ) goto etiket1;
}
```

goto

```
#include "stdio.h"
main( )
{
  printf("C'de");
  goto etiket 1;
  printf("Kitaplar");
  printf("Defterler");
  etiket 1:
  printf("goto deyiminin ");
  goto etiket 2;
  printf("Sayfalar");
  printf("Yapraklar");
  etiket 2:
  printf("Kullanılması");
}
```

goto

- Kod için kontrolsüzce atlamaya sebep olur
- Kullanılmaması önerilir.
- goto kullanmadan goto ile yapılabilecek her işlemin yapılabileceği ispatlanmıştır[Bohm 1966].

Döngülerle ilgili en iyi uygulamalar

* Döngü yazarken iç işlemden başlanması
* While ve for döngülerinden break veya return ile çıkılmaması
* Döngüden tek noktada çıkılması
* Sınırlar belirliyse while yerine for döngüsü tercih edilmesi
* For döngü indekslerinin tekrar tanımlanmaması
* Sınır değerlere dikkat edilmesi
* Döngü içerisinde statik işlemlerden kaçınılması

Döngü yazma sırası

* Döngü yazarken belirlenecekler.
  * İç işlem
  * Sınırlar veya Döngüye devam şartı
  * Eğer toplam alma gibi bir işlem yapılıyorsa sıfırlanma noktaları
* Öneri: Önce iç işlem belirlenir. Sonra döngü bunun etrafını saracak şekilde tasarlanır.

while ve for döngülerinden break veya return ile çıkılmaması

* Tek çıkış kontrolü kolaylaştırır.
* While ve for döngüyü sonlandıracak şartlara sahiptir. Özel durumlar hariç bu döngülerden çıkmak için break kullanılmamalıdır.

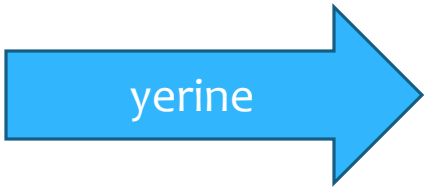
```
while (şart){
  if(şart){
     break;
  }
  …
 if(diğer_şart){
     break;
  }

}
```


```
while (şart){
  …
  …
  …
  …
 if(şart || diğer_şart){
     break;
  }

}
```

Döngülerden tek break veya return ile çıkılması

* Eğer break gibi şartlar kullanılmak istenirse tek bir noktadan döngüden çıkılmalıdır.
* Tek çıkış kontrolü kolaylaştırır.

```
while (şart){
  if(şart){
     break;
  }
  …
 if(diğer_şart){
     break;
  }

}
```

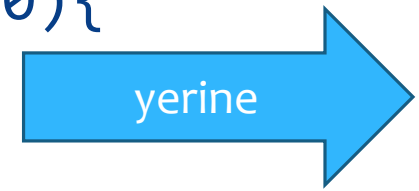



```
while (şart){
  …
  …
  …
  …
 if(şart + diğer_şart){
     break;
  }

}
```

Sınırlar belirliyse while yerine for döngüsü tercih edilmesi

* `while` nispeten karmaşıktır. Değişken kullanılacaksa ayrıca initialize edilmesi ve artma veya azaltma işlemi gerekir.
* Gereksiz kodun azalması yazılımda istenen bir özelliktir.

```
int i=0;
while (i<10){
 …
i++;
}
```




```
int i;
for(i=0;i<10;i++){
…
}
```

For döngü indekslerinin önceden tanımlanması

- for döngüsünde kullanılan indeks değişkenleri önceden tanımlanmalıdır.
- Aşağıdaki gibi bir kullanım **hatalıdır!**

```
for (int
i=0;i<10;i++){
…
}
```

Yukarıdaki kullanım java'da geçerlidir.

Sınır değerlere dikkat edilmesi

* Sınırlar en çok hata yapılan noktalardandır. Aşağıdaki döngüler kaç defa döner ? Deneyiniz.
* `for(int i=0;i<10;i++)`
* `for(int i=0;i<1;i++)`
* `for(int i=0;i<=1;i++)`
* `for(int i=5;i<=1;i++)`
* `for(int i=5;i<=1;i--)`

Döngü içerisinde statik işlemlerden kaçınılması

* Döngü içerisinde olması gerekmeyen atama vb. işlemler döngü dışına taşınmalıdır.

```
while (i<10){

int k=0;

i++;
}
```


```
int k=0;
while (i<10){

i++;
}
```

Ders Örnekleri

* Bir aralıktaki değeri çift elemanları toplatan bir program yazınız.
* Bir aralıktaki değeri çift ve tek eleman toplamları arasındaki farkı bulunuz.
* Bir aralıktaki değeri çift elemanların sayısını bulunuz.
* 5'ten 15 'e kadar olan sayıları toplamını for ve do while yapıları ile bulunuz.

Kontrol yapıları kullanılırken dikkat edilecekler: Tekrarlı if yerine else if kullanmak

* Eğer birden fazla bağlantılı kontrol kullanılacaksa tekrarlı if yerine else if yapısının tercih edilmesi

  Bu yapı iş mantığını açıkça yansıtır. İf şartlarından birisi doğru olursa diğerleri doğru olamaz.

```
Ara_deger=obje.hesapla()
 if (deger >Ara_deger)
 { i=10;}
Else if (deger > Ara_deger)
    {i=3;}
….
 if (deger > 2)
{//birşeyler çalıştır}
```

```
 if (deger == 1)
{//birşeyler çalıştır}
 else if (deger >= 1)
{//birşeyler çalıştır}
….
 else (deger == 1)
{//birşeyler çalıştır}
```

Sadece tek bir basit şartı test etmek

* Çok karmaşık if şartların anlamayı zorlaştırır. Bu yüzden mümkünse tek bir şart test edilmelidir.
* Gerekirse iç içe if ifadeleri kullanılabilir.
* Bu şekilde şartların tekrar yazmak gerekmez ve performans artabilir.

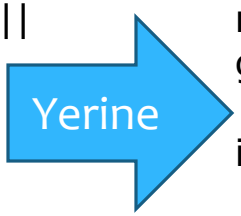
```
 if (deger == 1 && yil==2000)
{//birşeyler çalıştır}
 else if (deger >1 && yil==2000)
{//birşeyler çalıştır}
```


```
if (yil== 2000){
   if (deger==1) {
        //birşeyler çalıştır
   }
   else if (deger >1){
        //birşeyler çalıştır
   }
}
```

Şartı test için boolean değişkenler kullanmak

* Çok karmaşık if şartların anlamayı zorlaştırır. Bu yapıları mümkünse tek bir şart test edilmelidir.
* Bu şekilde de şartların tekrar yazmak gerekmez ve performans artabilir.

```
 if (maas < 1000 && giris_yili<=2000 ||
unvan= "MEMUR")
 {//maas arttisi için birşeyler çalıştır}
```




```
boolean maasArtisinaUygun;
maasArtisinaUygun= (maas < 1000 &&
giris_yili<=2000 || unvan= "MEMUR") ;

if (maasArtisinaUygun){
        // maas arttisi için birşeyler çalıştır
}
```

Derin Hiyerarşilerden kaçınmak

* İç içe kontrol ve döngü ifadeleri kodu anlamayı zorlaştırır.
* Derinliğin 3'ü geçmemesi tavsiye edilir.
* Bu yapılar ayrı metotlara taşınmalıdır.

```
if (şart) {
   if  (şart){
    for(..;..;..) {
        while (şart){
        //birşeyler çalıştır
        }
   }
}
}
```

Yerine

```
if (şart) {
   if  (şart){
      // metod çağır
}
}
```

```
metod{
for(..;..;..) {
        while (şart){
        //birşeyler çalıştır
        }
   }
}
```

Örnekler

* İki sayıdan büyük olanı bulan bir yazılım tasarlayınız.
* Üç sayıdan en büyüğünü bulan bir yazılım tasarlayınız.
* Dört sayıdan en büyüğünü bulan bir yazılım tasarlayınız.
* Haftanın günlerinin numaraları girildiğinde ismini veren bir yazılım tasarlayınız. Pazartesi sıfır kabul edilecektir.
* Bir değişken değerinin tek mi çift mi olduğunu ve 10'dan büyük mü küçük mü olduğunu ekrana yazan bir program tasarlayınız.