

BİLGİSAYAR PROGRAMLAMA

Algoritma ve Akış Şemaları

Konu Başlıkları

- Algoritma tanımı
- Algoritma özellikleri
- Algoritma tasarımı
- Akış şemaları
- Dallanma simgeleri
- Döngü simgeleri
- Akış şeması tasarımı
- Akış şeması örnekleri

Algoritma

- Herhangi bir işi yapmak ya da bir problemi çözmek için adım adım uygulanan kurallar dizisine algoritma denir.
- Algoritmada adımlar sırasıyla işlenmektedir.

Algoritma

- Bir algoritmanın en önemli özellikleri:
 - bir giriş verisine karşılık çıkış bilgisinin mutlaka bulunması,
 - sonlu sayıda adım içermesi,
 - her türlü alternatifin düşünülerek sonuca ulaşıldığının garanti edilmesidir.
- Eğer sonlu sayıda adım ile Sonuca ulaşılamaz ise istenmeyen Sonsuz döngüler, eğer bütün alternatifler düşünülmez ise belirsiz durumlar ortaya çıkabilir ve bir sonuca ulaşılamaz.

Algoritma

- Bir kurallar dizisini algoritma olarak tanımlayabilmek için mutlaka giriş verileri değerlendirilerek çıkış bilgisinin elde edilmesi gereklidir.
- Algoritma, belirli biri işin/problemin sonucunu elde etmek için art arda uygulanacak adımları ve koşulları kesin-olarak ortaya koyar. Bu adımlar, ilgili koşullar altında adım adım izlendiğinde bir sonuca ulaşılır.

Algoritmada olması gereken özellikler

- **Etkin ve Genel olma:** her koşulda ve her giriş değerinde doğru sonuca ulaşılabilmelidir.
- **Sonlu olma.** Algoritma kesinlikle sonlu sayıda işlem içermeli ve bu işlemlerin süresi de sonlu olmalıdır;
- **Yanılmazlık.** Algoritma tekrar yürütüldüğünde aynı giriş değerleri için aynı sonuç elde edilmelidir.
- **Giriş/Çıkış tanımlı olma.** Algoritmanın giriş ve Çıkış değerleri olmalıdır.
- **Başarım.** Algoritma başarımı (disk ve bellek kullanımı) iyi olacak şekilde tasarlanmalıdır.

Akış Şeması

- Sorunun çözümü için oluşturulmuş algoritmanın görsel olarak şekillerle ifade edilmesine akış şeması (flow chart) denilmektedir.
- Akış şemalarında algoritmanın adımları simgeler şeklinde kutular içine yazılmaktadır ve adımlar arasındaki ilişkiler ve akış yönü oklar ile gösterilmektedir.

Akış Şemasının Avantajları

- 1) Birbiri ile ilgili adımlar arasındaki mantıksal ilişkiyi gösterirler.
- 2) İzlenmesi ve anlaşılması kolaydır.
- 3) Şartlara bağlı olarak ortaya çıkan olayların takip edilmesi mümkündür.
- 4) Belli bir standarda göre hazırlandıklarından herkesçe anlaşılabilir ve birden fazla kişi aynı şema üzerinde çalışabilir.

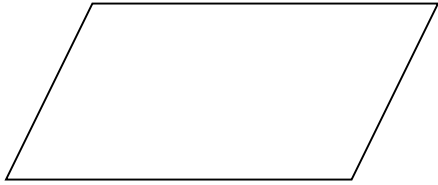
Akış Şeması Hazırlama Kuralları

- 1) Başlangıç ve bitiş uçları tanımlanmalıdır.
- 2) Standart semboller kullanılmalıdır.
- 3) Birbirini kesen akış hatları kullanılmamalıdır.
- 4) Basit kararlar alınmalıdır.
- 5) Şema belli bir yönde hazırlanmalıdır.

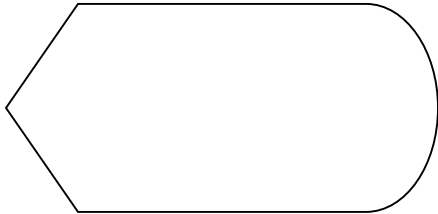
Akış Şeması Elemanları



Akış şemasının başlangıcını yada bitişini belirtir.

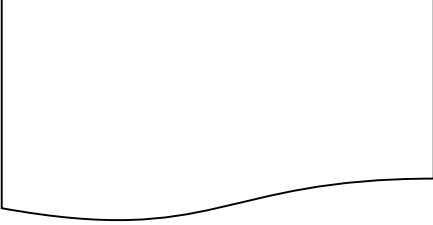


Veri girişi yapılacağını belirtir.



Ekranda görüntüleme yapılacağını belirtir.

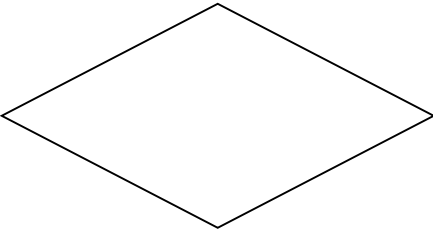
Akış Şeması Elemanları



Yazıcıya çıktı olacağını belirtir.



Hesaplama ya da değerlerin değişkenlere aktarımını gösterir.

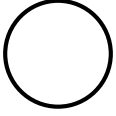


Aritmetik ve mantıksal ifadeler için karar verme ya da karşılaştırma durumunu gösterir.

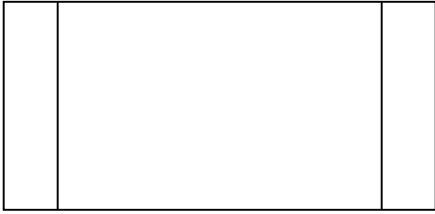
Akış Şeması Elemanları



Yapılacak işler birden fazla sayıda yinelenecek ise yani iş akışında döngü var ise bu sembol kullanılır.



İki nokta arası ilişkiyi gösterir. Döngü sonunu göstermek için ya da akış şemasının başka bir yere bağlantısını göstermek amacıyla kullanılır.



Fonksiyon çağırılacağını belirtir.



İş akışının yönünü belirtir.

Akış Şemaları

- Akış şemaları içerik ve biçimlerine göre genel olarak üç grupta sınıflandırılabilirler.
 1. Doğrusal akış şemaları
 2. Mantıksal akış şemaları (Koşullu)
 3. Döngü içeren akış şemaları

Doğrusal akış şemaları

- İş akışları, giriş, hesaplama, çıkış biçiminde olan akış şemaları bu grup kapsamına girer.
- Yapısında karar alma ya da döngü ifadeleri içermeyen akış şemalarıdır.

Örnek-1

(Ekranı Merhaba Dünya Yazdırma)

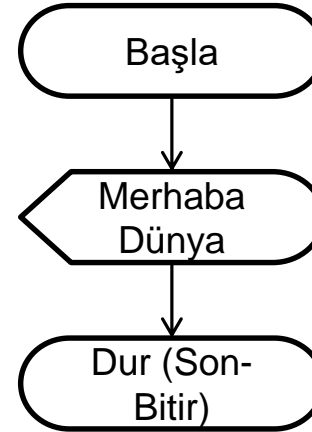
Algoritması

Adım 1: Başla
Adım 2: Ekranı 'Merhaba Dünya' yazdır.
Adım 3: Dur

C Kodu

```
#include <stdio.h>
void main()
{
    printf("Hello World");
}
```

Akış Şeması (doğrusal)



Örnek-2

(İki Sayıyı Toplama ve Görüntüleme)

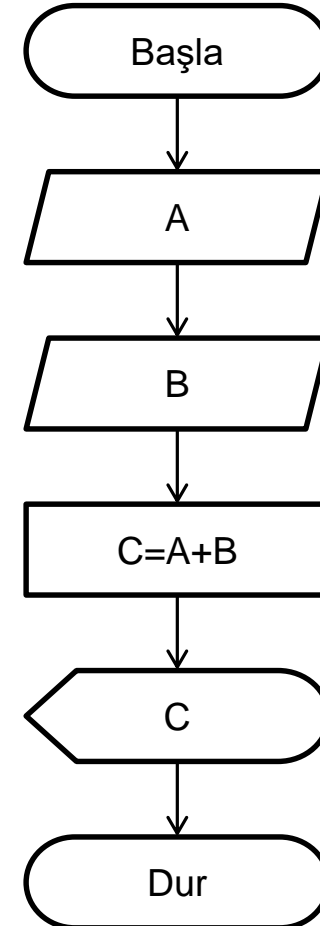
Algoritması

Adım 1: Başla
Adım 2: Birinci sayıyı gir
Adım 3: İkinci sayıyı gir
Adım 4: İki sayıyı topla
Adım 5: Toplam sonucunu ekranda görüntüle
Adım 6: Dur

Değişkenler:
A: Birinci sayı
B: İkinci sayı
C: Toplam sonucu

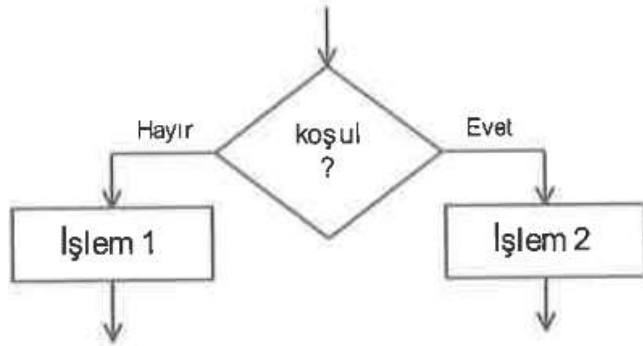
Adım 1: Başla
Adım 2: A'yı gir
Adım 3: B'yi gir
Adım 4: $C=A+B$
Adım 5: C'yi ekranda görüntüle
Adım 6: Dur

Akış Şeması (doğrusal)

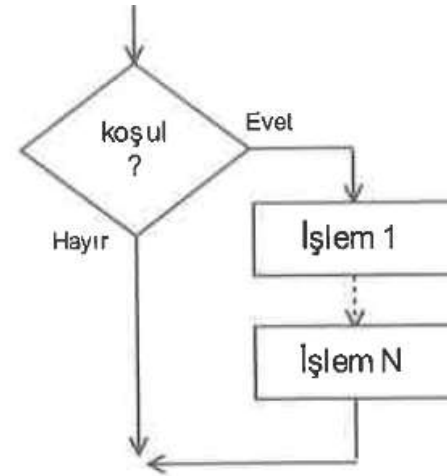


Mantıksal (Koşullu Dallanma)

- Geniş ölçüde mantıksal kararları içeren akış şemalarıdır.
- Hesap düzenleri genellikle basittir.
- Verilen koşulun doğru yada yanlış olmasına göre iş akışı yönlendirilir.



a) Koşulun durumuna bağlı iki farklı işlem var



a) Olumsuz koşulda yapı alacak yok; olumluda N tane işlem var

Örnek-3

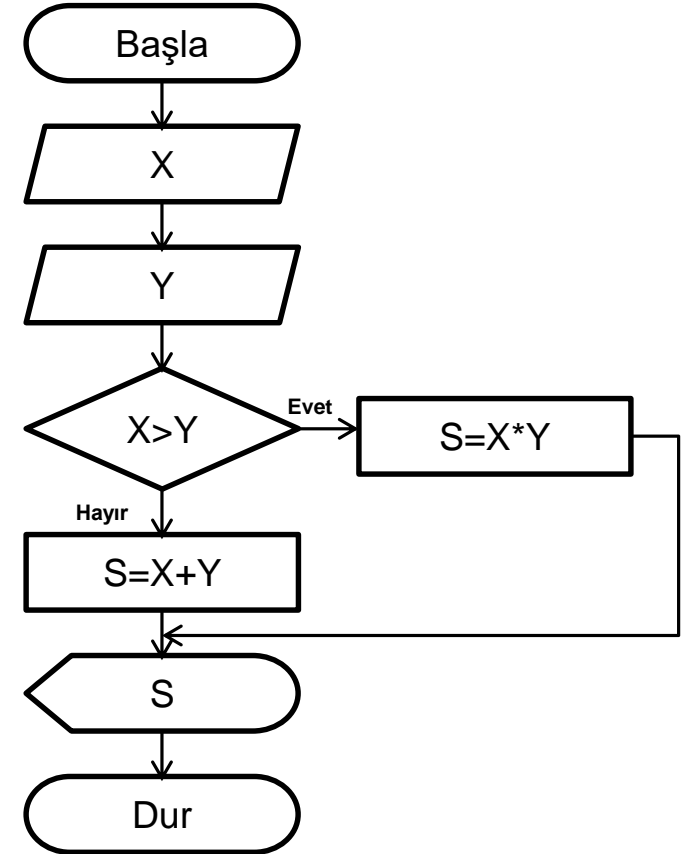
(Klavyeden Girilen 2 sayıdan birincisi büyük ise çarpma, aksi durumda toplama yapan algoritma ve akış şeması)

Algoritması

Değişkenler:
X: Birinci sayı
Y: İkinci sayı
S: Toplam sonucu

Adım 1: Başla
Adım 2: X'i gir
Adım 3: Y'yi gir
Adım 4: $X > Y$ ise $S = X * Y$ hesapla adım 6'ya git
Adım 5: $X \leq Y$ ise $S = X + Y$ hesapla
Adım 6: S'i görüntüle
Adım 7: Dur

Akış şeması



Örnek: İki sayıdan büyük olanı bulma

Algoritma bulBuyuk;

/ Bu algoritma iki sayı okur ve büyük olanı bulur; sonucu ekrana yazdırır. */*

❶ **Oku (Sayı A) ;** */* birinci sayıyı oku */*

❷ **Oku (Sayı B) ;** */* ikinci sayıyı oku */*

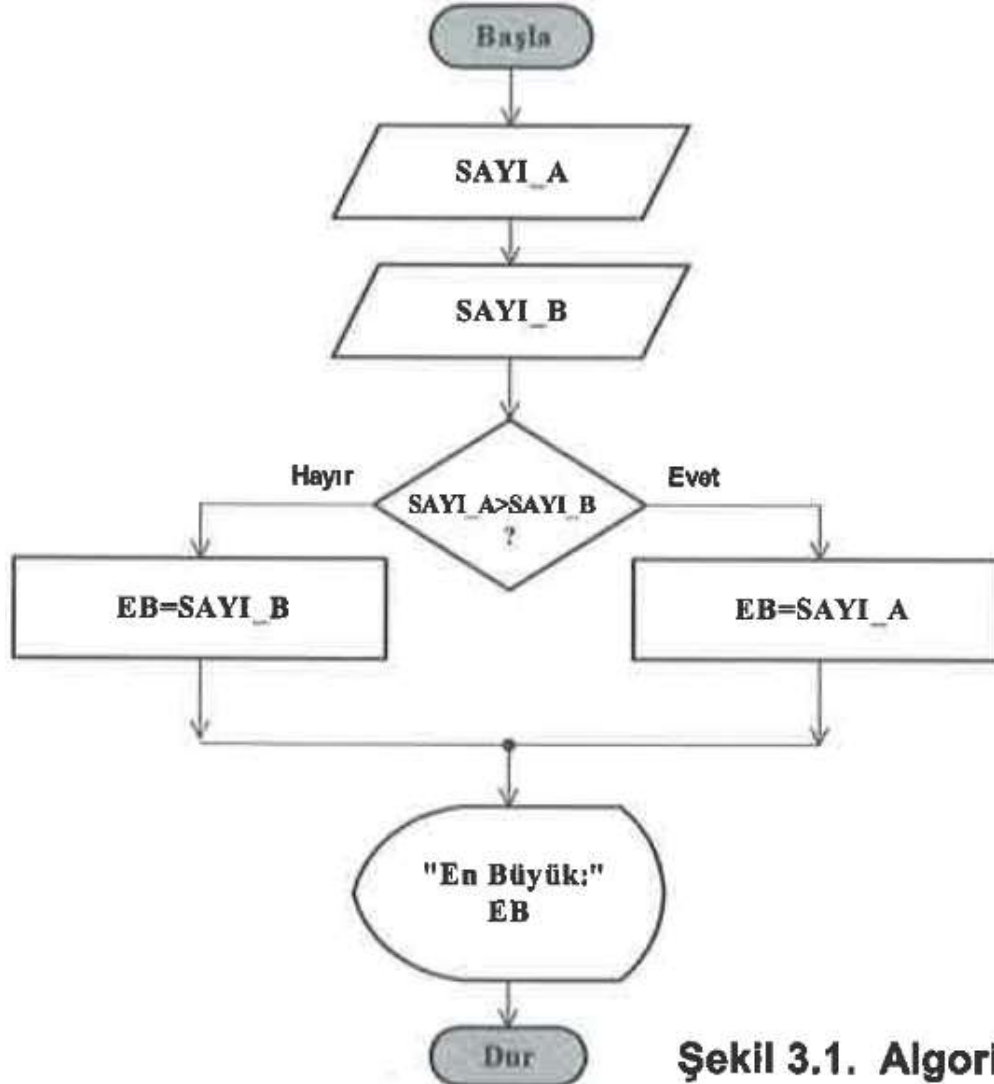
❸ **Karşılaştır (A>B) ?** */* sayıları karşılaştır; A, B'den büyük mü? */*
 Evet ➡ EB=A
 Hayır ➡ EB=B

❹ **Yaz (EB) ;** */* büyük olan EB'de bunu ekrana yazdır */*

❺ **Dur ;**

Algoritma 3.1. İki sayıdan büyüğünü bulan algoritma

Örnek: İki sayıdan büyük olanı bulma



Şekil 3.1. Algoritma 3.1 in akış şeması

Döngüsel Akış Şemaları

- Sorunun çözümü için, çözümde yer alan herhangi bir adım ya da aşamanın birden fazla kullanıldığı akış şemalarına denir.
- İş akışları genel olarak giriş ya da başlangıç değeri verme, hesaplama, kontrol biçiminde olmaktadır.

Örnek-4

(Klavyeden 5 kişinin doğum yılını girip 2013 yılındaki yaşını hesaplayan algoritma ve akış şeması)

Algoritması

Değişkenler:

dy : kişinin doğum yılı

yas: kişinin yaşı

i : sayaç (5 kişi olup olmadığını kontrol eder)

Adım 1: Başla

Adım 2: $i=1$;

Adım 3: dy'yi gir

Adım 4: $yas=2013-dy$

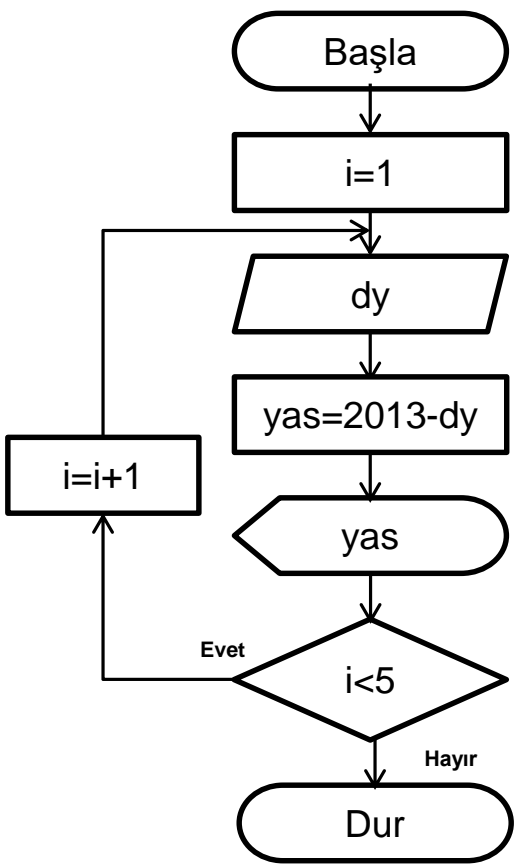
Adım 5: yas'ı ekrana yazdır

Adım 6: $i<5$ ise i'yi 1 arttır, adım 3'e git

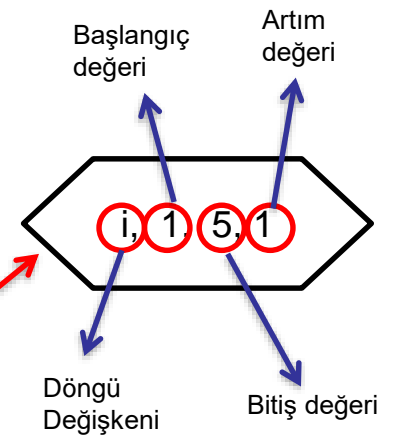
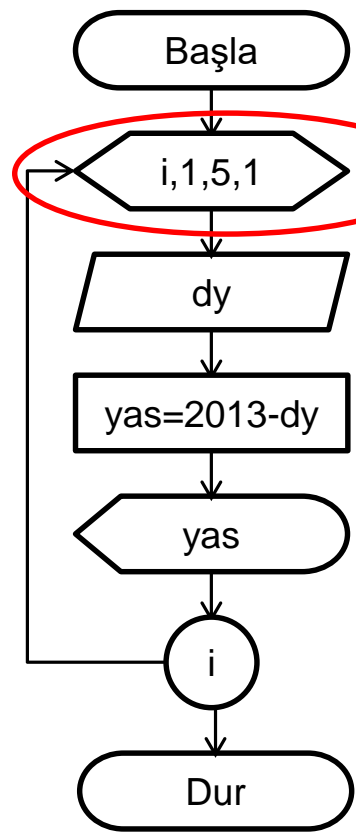
Adım 7: Dur

Örnek 4 (Devam)

Akış şeması I
(Döngü ifadesi içermeyen)



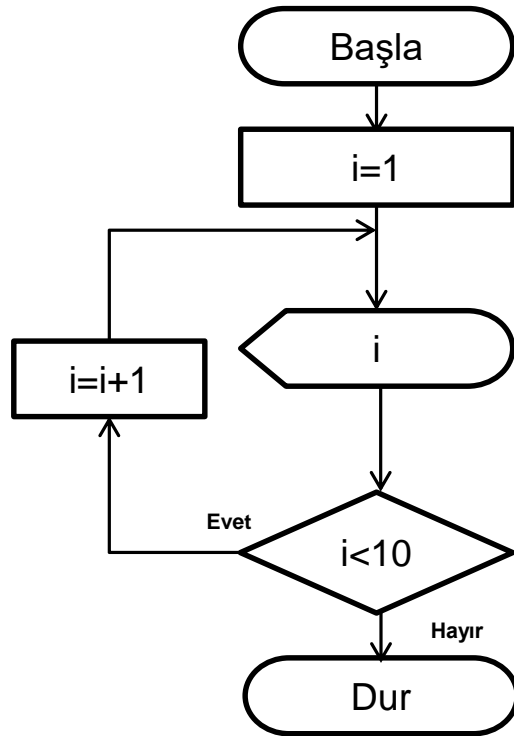
Akış şeması II
(Döngü şekli içeren)



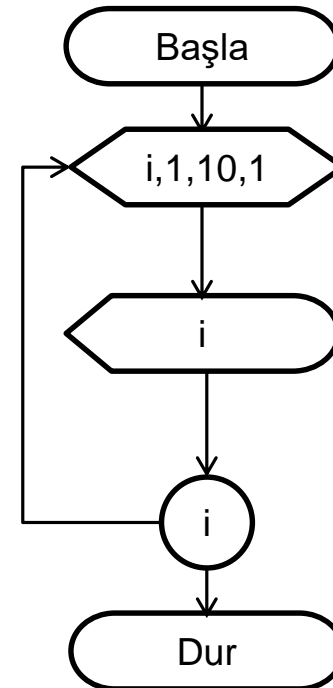
Örnek 5

(1'den 10'a kadar sayıları ekranda görüntüleyen akış şeması)

Kontrol İfadesiyle



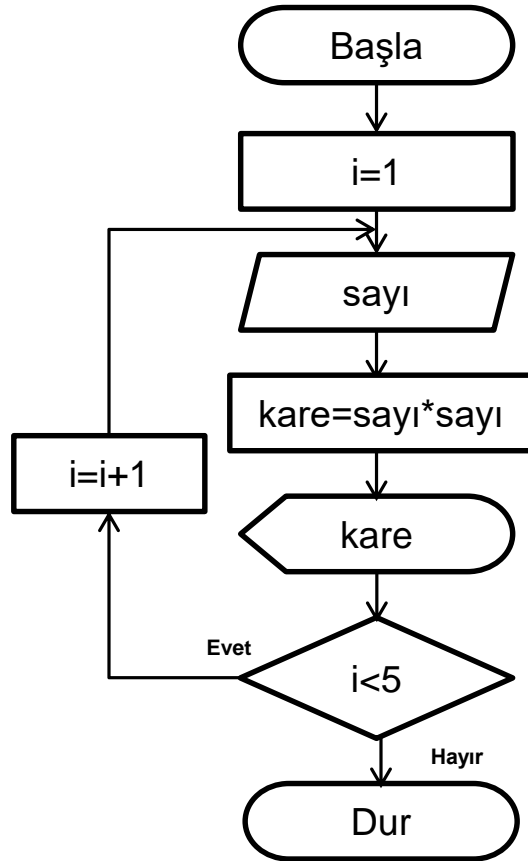
Döngü İfadesiyle



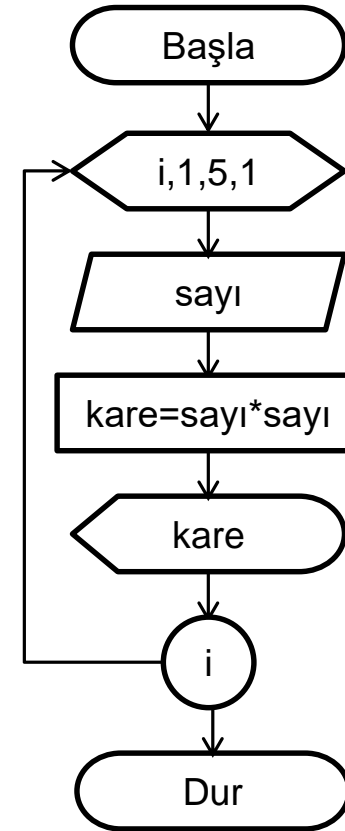
Örnek 6

(Girilen 5 sayının karelerini görüntüleyen akış şeması)

Kontrol İfadesiyle



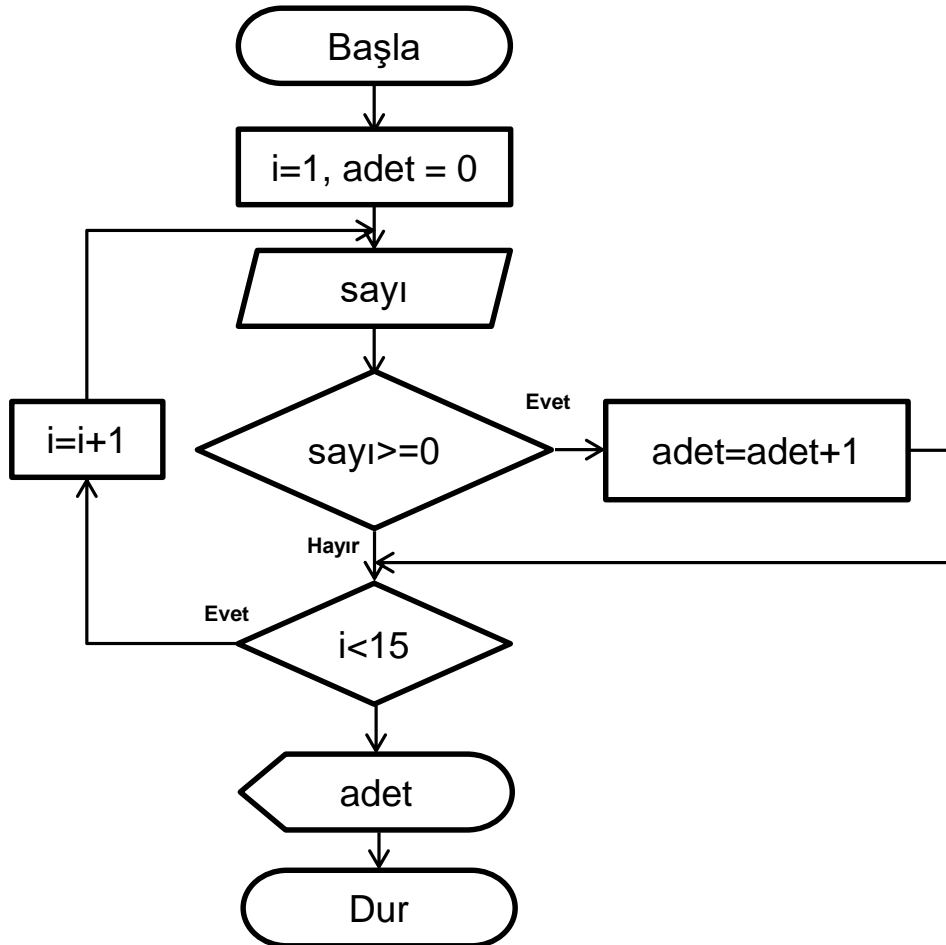
Döngü İfadesiyle



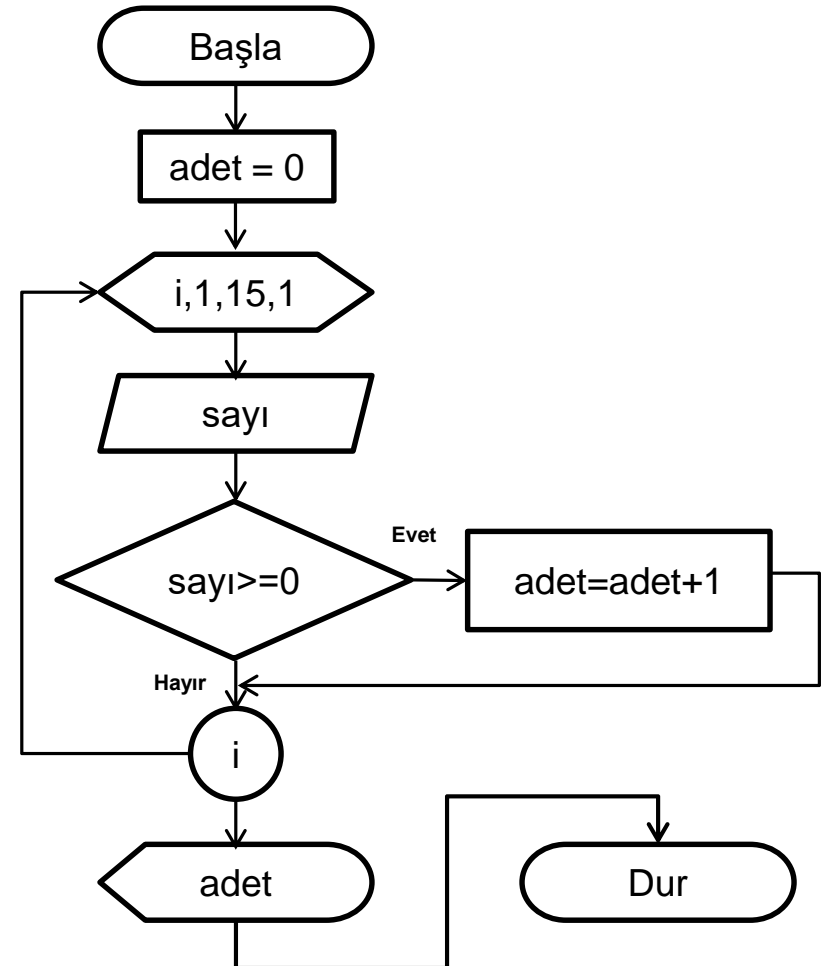
Örnek 7

(Girilen 15 sayıdan pozitif olanların adedini bulup görüntüleyen akış şeması)

Kontrol İfadesiyle



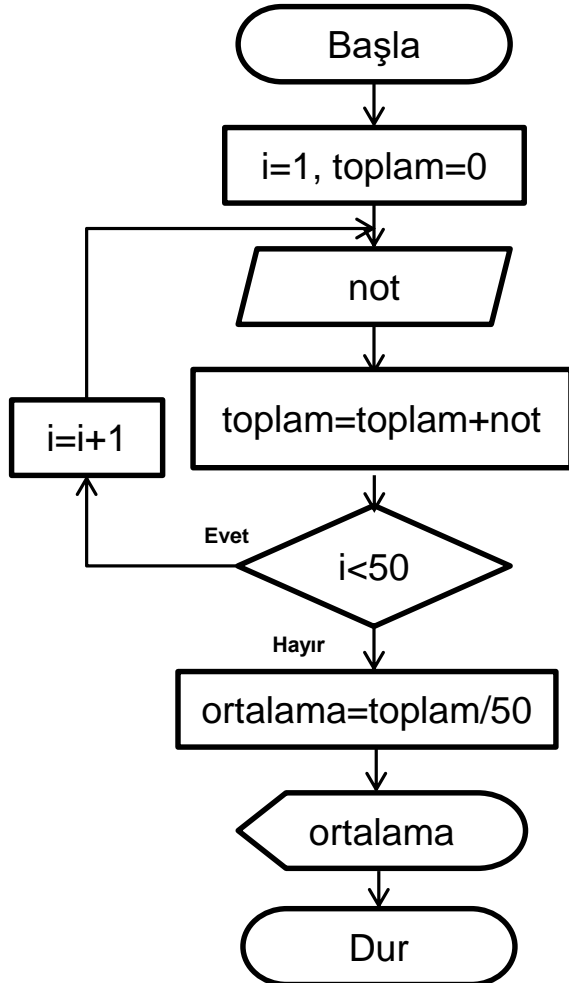
Döngü İfadesiyle



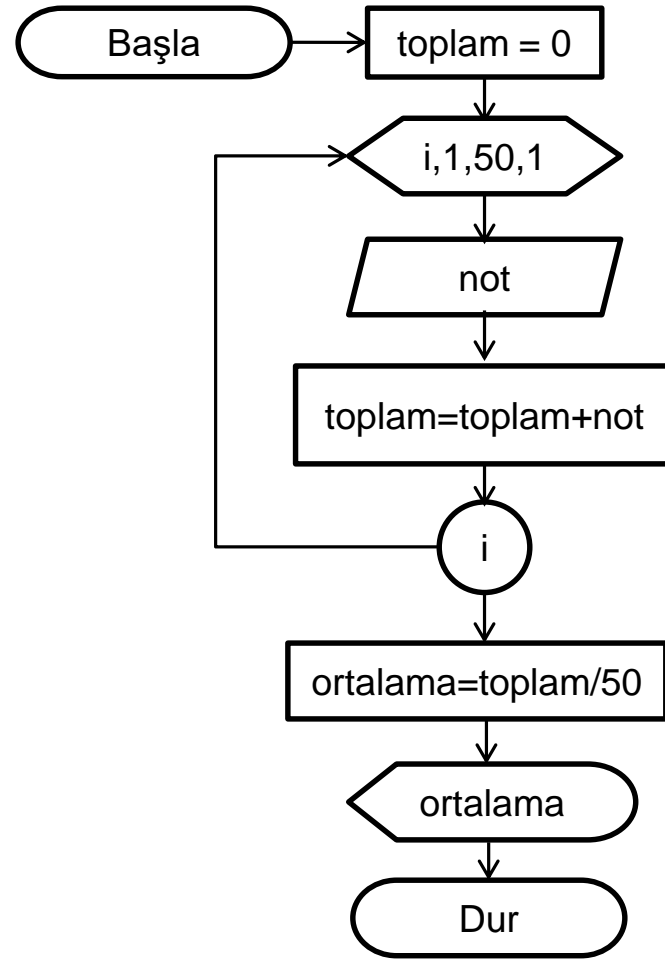
Örnek 8

(50 öğrencinin notlarının ortalamasını bulan akış şeması)

Kontrol İfadesiyle



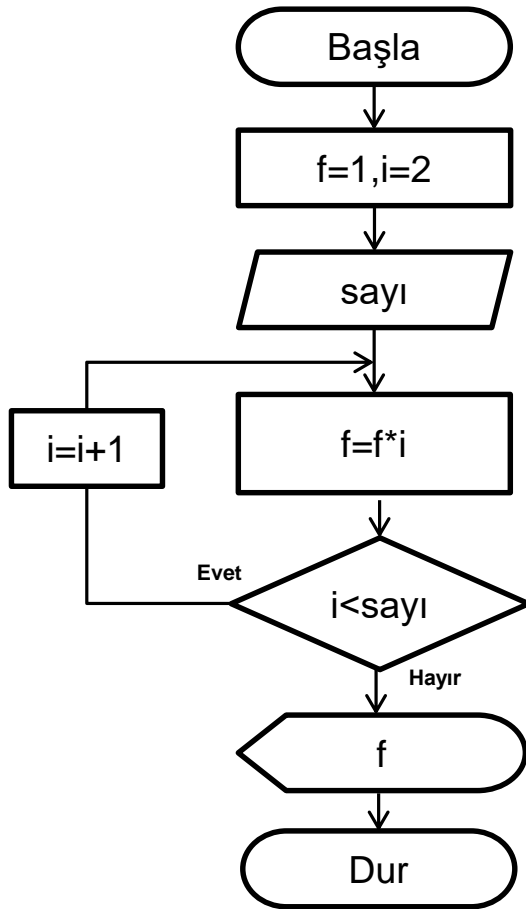
Döngü İfadesiyle



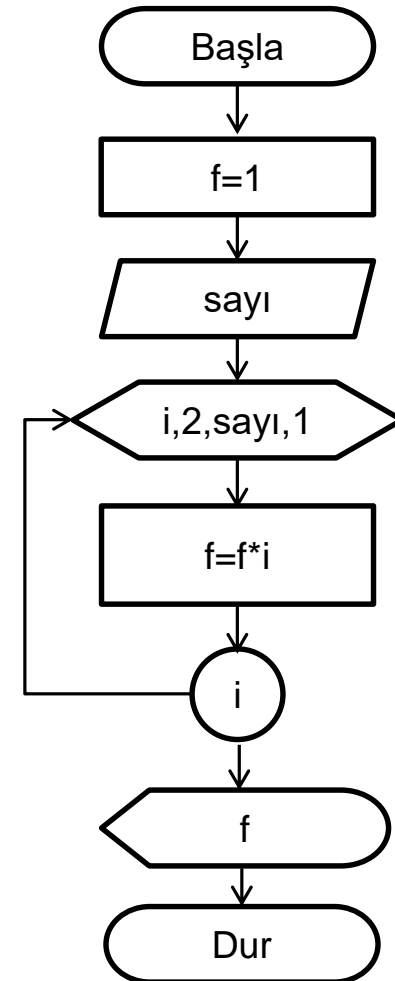
Örnek 9

(Girilen sayının faktöriyelini hesaplayan akış şeması)

Kontrol İfadesiyle



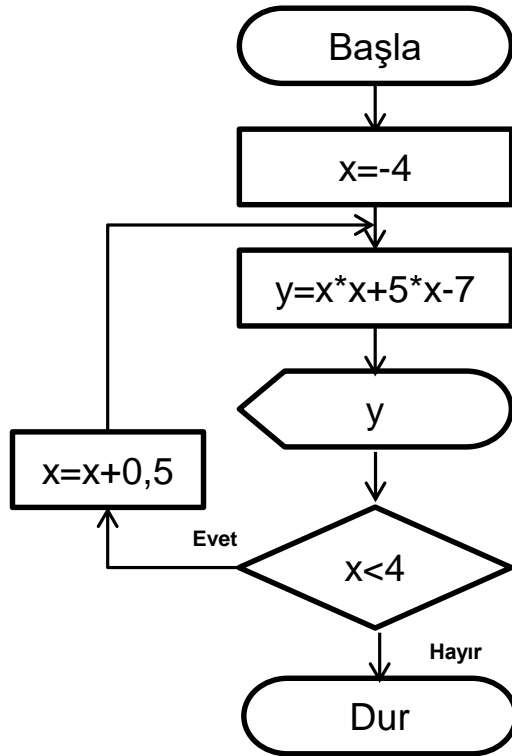
Döngü İfadesiyle



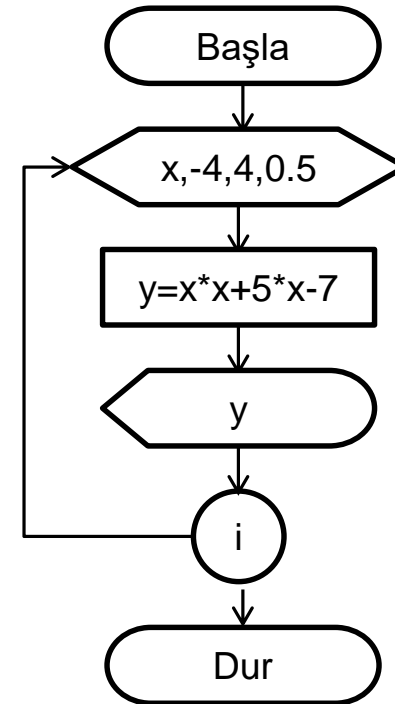
Örnek 10

($y=x^2+5x-7$ denkleminin $x=[-4,4]$ aralığındaki çözümlerini bulan ve görüntüleyen akış şeması, (x 'in artım değeri 0,5'tir.))

Kontrol İfadesiyle



Döngü İfadesiyle

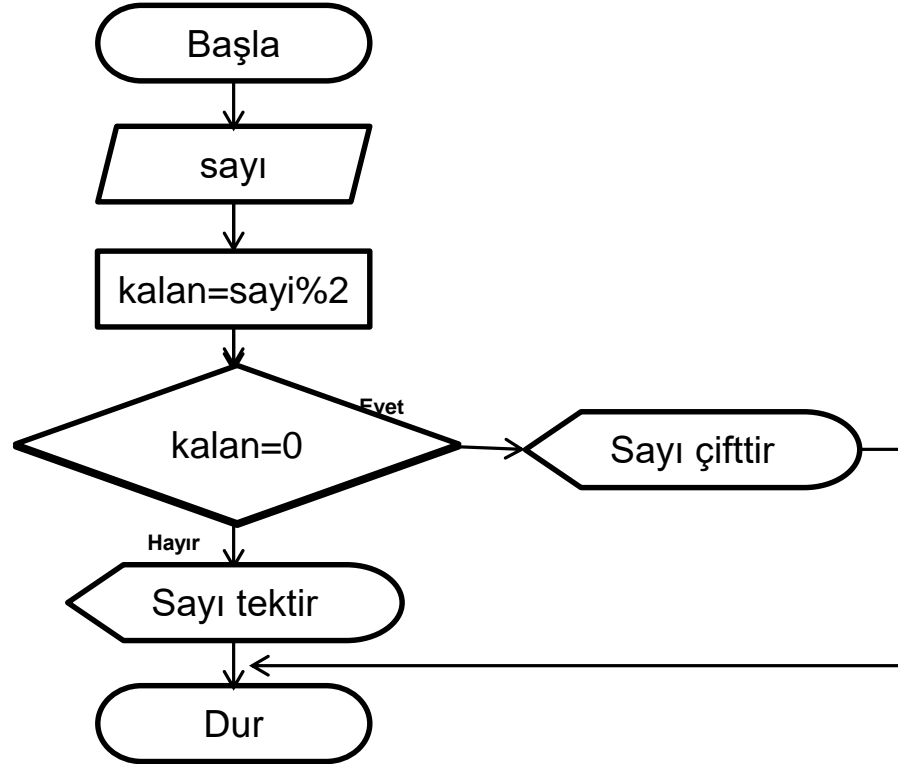


Örnek 11

(girilen sayının tek yada çift olduğunu bulup uygun mesajı görüntüleyen akış şeması)

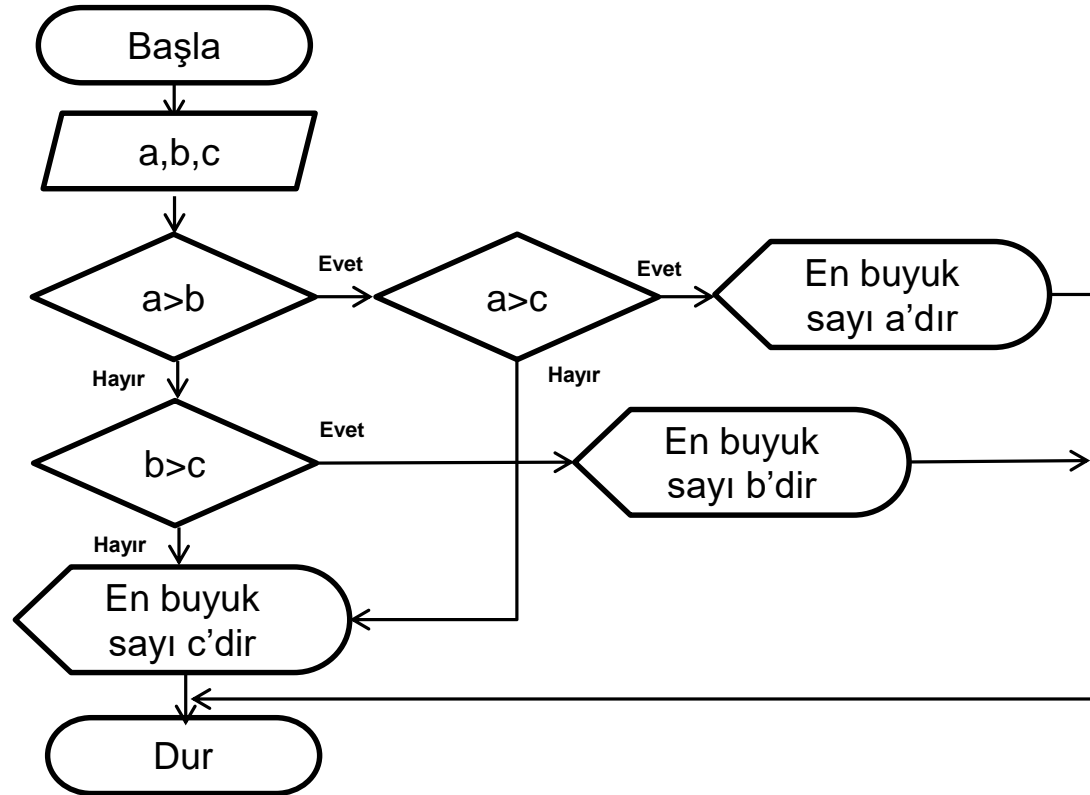
% simgesi mod alma işlemidir. $\text{sayi mod } 2 = 0$ ise çift sayı olduğunu değilse tek sayı olduğunu ekrana gösteriyoruz.

kalan adında bir değişken kullanmadan, kontrol ifadesi içinde $\text{sayi \% } 2 = 0$ yazmamız da mümkün olabilirdi.



Örnek 12

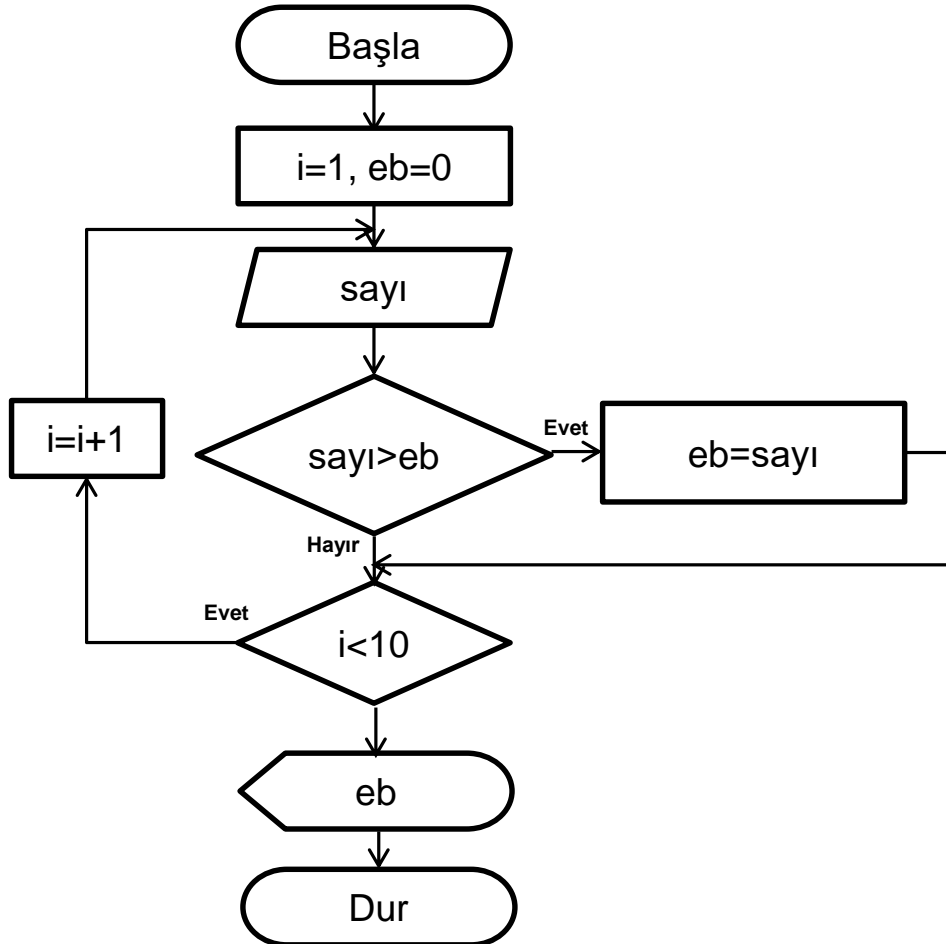
(girilen 3 sayıdan hangisinin en büyük olduğunu bulan akış şeması)



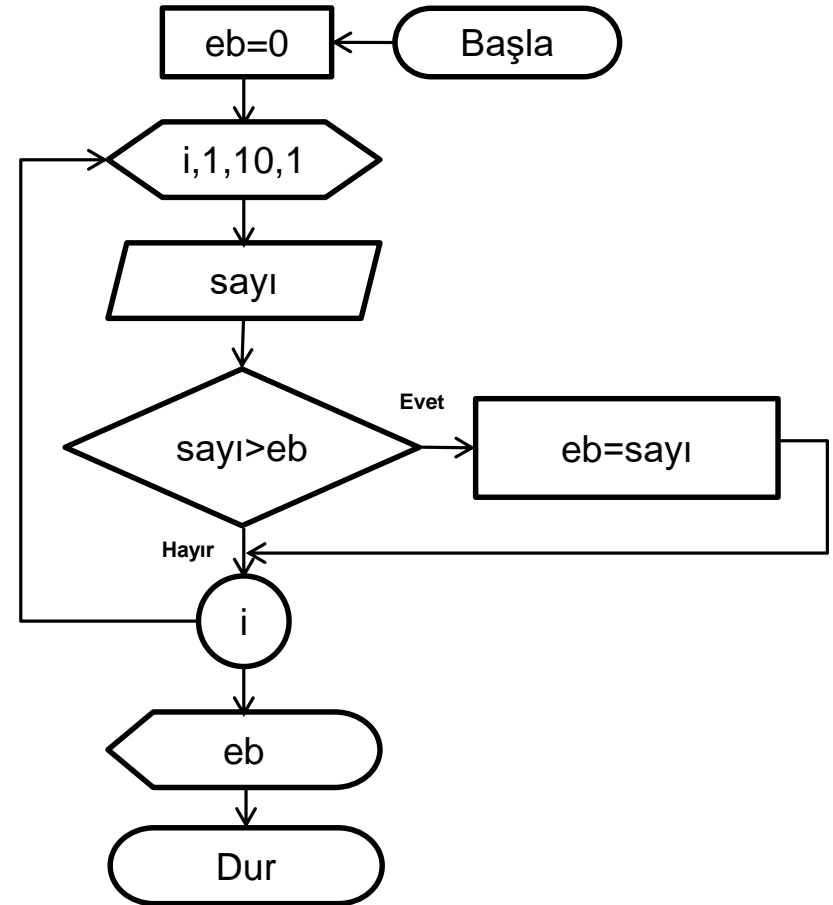
Örnek 13

(girilen 10 sayıdan en büyüğünü bulan ve görüntüleyen akış şeması)

Kontrol İfadesiyle



Döngü İfadesiyle



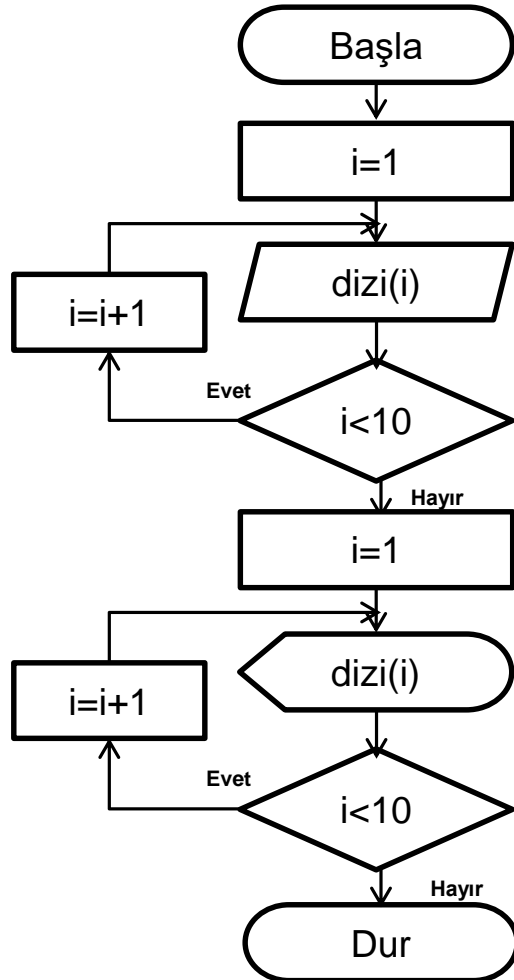
Dizi Kavramı

- Diziler aynı tipteki elemanların yan yana sıralanmasıyla elde edilen bir bilgi kümesidir.
- İki boyutlu (örn: matris) yada daha çok boyutlu diziler de olabilir.
- Diziler akış şemalarında indisli gösterilirler:
 - $dizi(0)$, $dizi(1)$, $dizi(2)$,, $dizi(i)$
 - ilk elemanın indisi akış şemalarında 1 olarak gösterilse de, C programlama dilinde 0'dır.

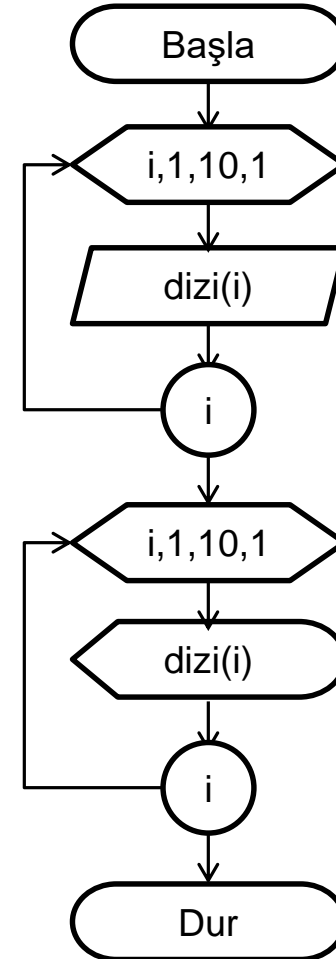
Örnek 15

(10 elemanlı bir diziye bilgi girişi yapan ve diziyi görüntüleyen akış şeması)

Kontrol İfadesiyle

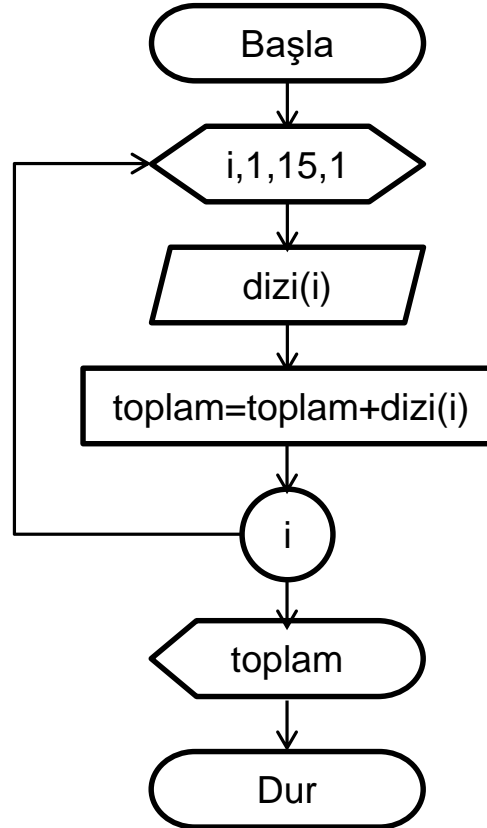


Döngü İfadesiyle



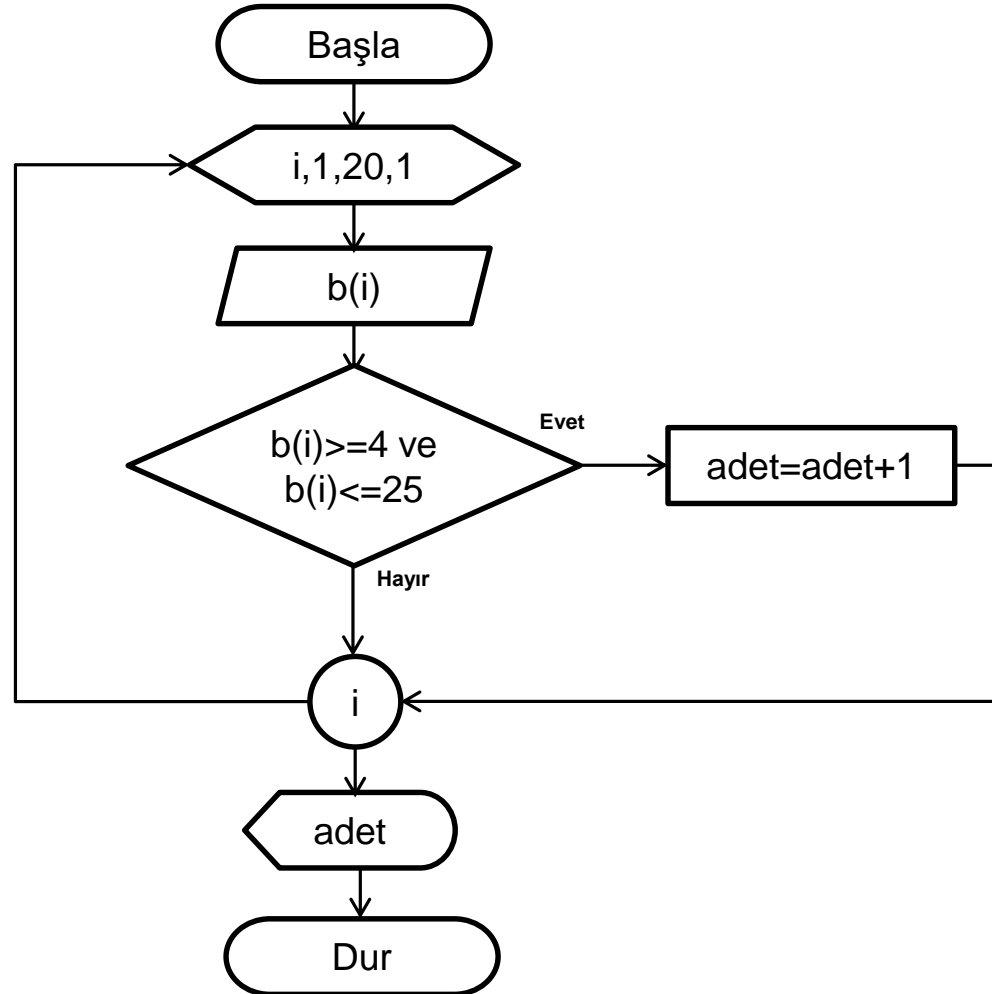
Örnek 16

(15 elemanlı bir sayı dizisine bilgi girişi yapılarak elemanların toplamını bulan akış şeması)



Örnek 17

(20 elemanlı bir sayı dizisine bilgi girişi yapılarak 4 ile 25 arasında olanların adedini bulan ve görüntüleyen akış şeması)



Kaba Kod (Pseudo Code) (Sözde Kod)

- Kaba-kod, algoritmanın yarı programlama dili kuralları, yarı konuşma dili metinleri ile ortaya koyulması/tanımlanmasıdır.
- Gerçek kod ise, algoritmanın herhangi bir programlama diliyle gerçekleştirilmiş halidir.
- Kaba-kod ile verilen bir problem yazılımcılar tarafından istenen bir Programlama dili ile kodlanabilir.
- Kaba-kod herhangi bir programlama dilinden bağımsız olup herhangi bir kurala da sahip değildir.

Kaba Kod Örneği

Örnek 3.4. Girilen iki sayıdan büyük olanını belirleyen ve onu ekrana yazdıran algoritmayı tasarlayınız.

Algoritma büyükOlanSayı

```
/* Bu algoritma girilen iki sayıdan büyük olanını belirler ve yazdırır */
```

```
❶ Oku (A)      /* 'A' Değerini Oku.*/
```

```
❷ Oku (B)      /* 'B' Değerini Oku.*/
```

```
❸ if A<B then Eb←B      /* 'A' değeri ile 'B' değerini kıyasla büyük olanını 'Eb' ye  
   else Eb←A           ata. */
```

```
❹ Yaz (Eb) ;      /* 'Eb' değerini yazdır.*/
```

```
❺ Dur;           /* İşlem bitir.*/
```

Algoritma 3.4. Alternatif koşullu 'If' ifadesini işleyen algoritma

Kaba Kod Örneği

Algoritma EnKucukBul;

/* Bu algoritma N elemanlı bir A dizinin en küçük değerli elemanını bulur. */

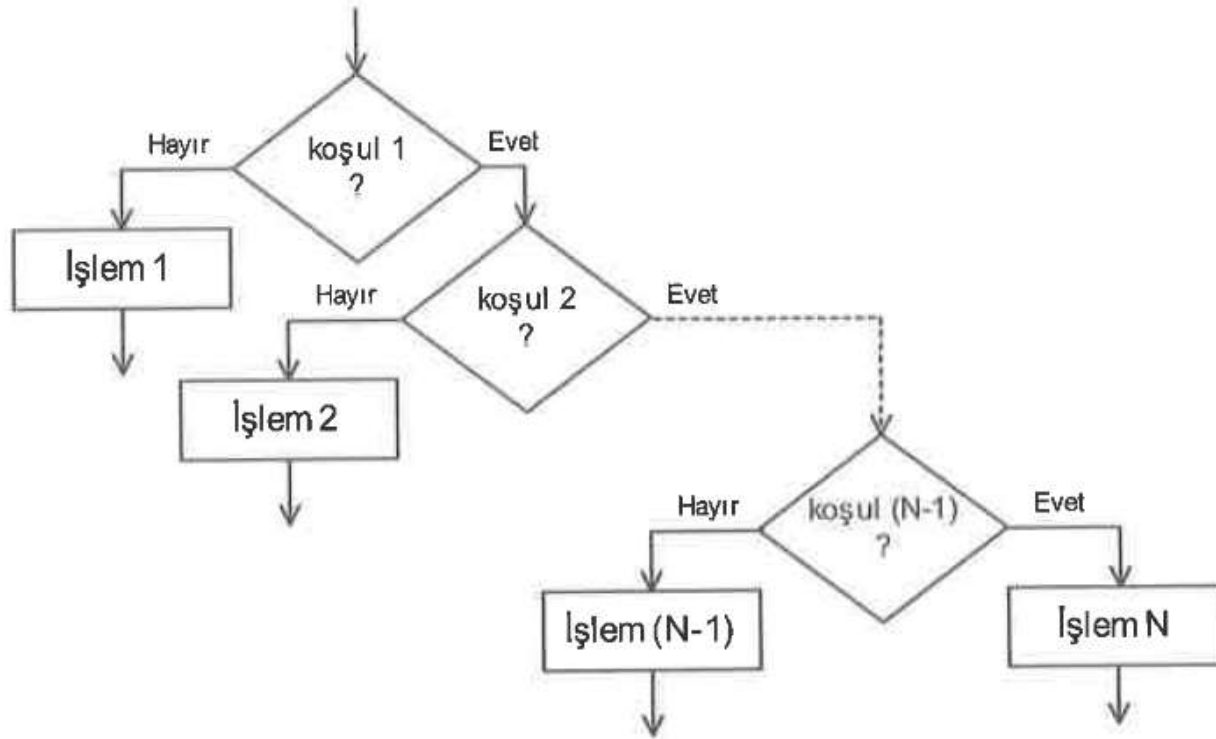
Adım

- ①. [Dizi boş mu?] /* dizide eleman olup olmadığı sınanır */
if $N < 1$ then Begin
 Yaz('Dizide eleman yok');
 Geridön;
- ②. [İlk İşlemler]
EKE ← A[1]; /* Başlangıçta dizinin ilk elemanı en küçük kabul edilir.*/
i ← 2; /* İkinciden başlanacağı için dizi indis değeri 2 olmaktadır.*/
- ③. [Dizinin elemanlarını EKE ile karşılaştır]
i ≤ N olduğu sürece 5. adım dahil tekrarla
- ④. [Dizinin bir sonraki elemanı ile EKE ile karşılaştır]
if $A[i] < EKE$ then $EKE ← A[i]$; /* Dizinin i. elemanı daha küçükse onu EKE kabul et. */
- ⑤. [indis değerini bir artır.]
i ← i + 1;
- ⑥. [İşlemi Bitir]
Geridön;

Algoritma 3.2. Dizinin en küçük değerli elemanını bulan algoritma

İç İçe karşılaştırmalı ifadeler -1 (nested statements)

Belirli bir koşul göre işlem yaptırılacaksa, eğer koşula göre iki farklı işlem varsa bir karşılaştırma işlemiyle yapılabilir. Ancak, koşullara göre birden ikiden çok seçenek varsa karşılaştırma deyimleri içice kullanılabilir. İçice *if* cümlesi yapılarından ikisi aşağıda görüldüğü gibidir:



İç içe if (C-örneği)

```
#include <stdio.h>

int main () {

    /* local variable definition */
    int a = 100;
    int b = 200;

    /* check the boolean condition */
    if( a == 100 ) {
        /* if condition is true then check the following */
        if( b == 200 ) {
            /* if condition is true then print the following */
            printf("Value of a is 100 and b is 200\n" );
        }
    }
    return 0;
}
```

İç İçe karşılaştırmalı ifadeler -2 (case statements)

Durum İfadesi

CASE

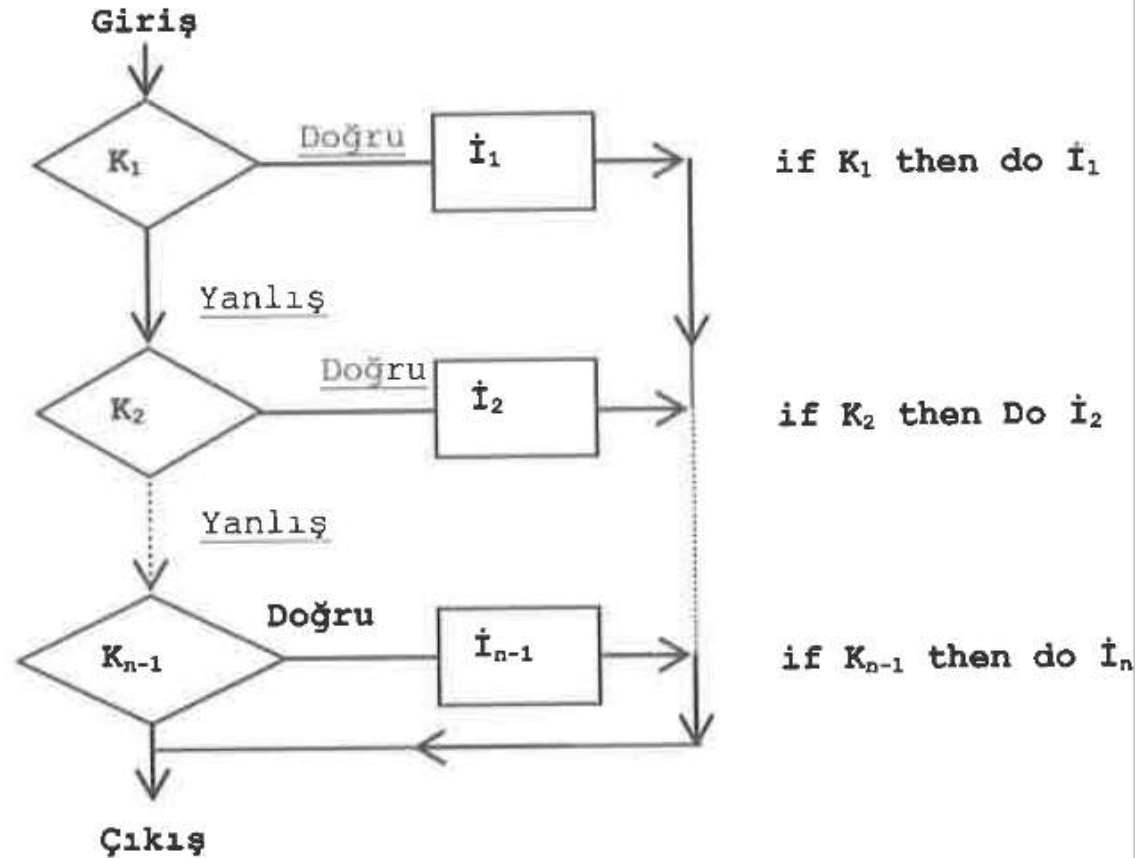
:<koşul₁(k₁)>:İ₁

:<koşul₂(k₂)>:İ₂

⋮

:<koşul_{n-1}(k_{n-1})>:İ_{n-1}

End



İç içe karşılaştırma – case (C-örneği)

```
#include <stdio.h>
int main () {

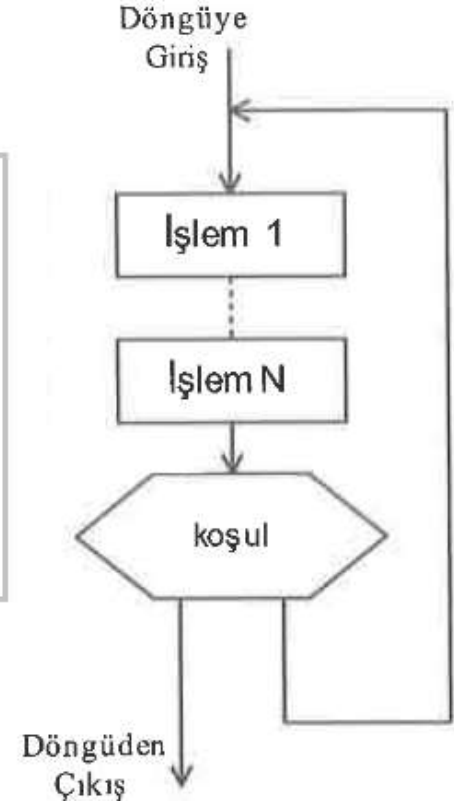
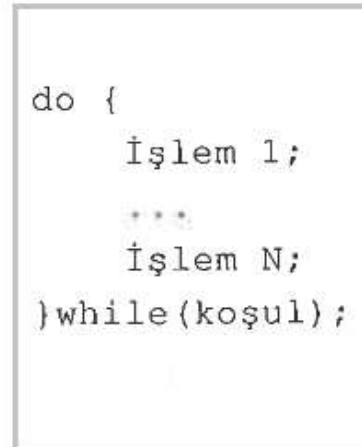
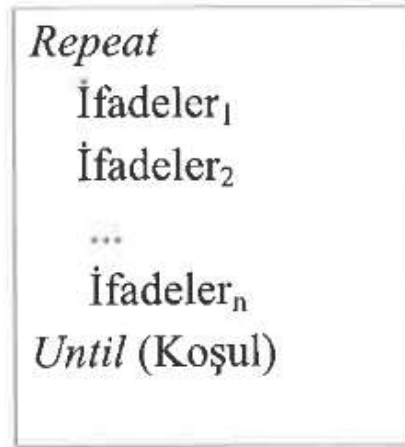
    /* local variable definition */
    char grade = 'B';

    switch(grade) {
        case 'A' :
            printf("Excellent!\n" );
            break;
        case 'B' :
            printf("Well done\n" );
            break;
        case 'C' :
            printf("You passed\n" );
            break;
        case 'F' :
            printf("You failed.\n" );
            break;
        default :
            printf("Invalid grade\n" );
    }
    return 0;
}
```

Döngü Çeşitleri

1. Repeat-Until (Do-while) Döngüleri

Repeat döngüsünde koşul ne olursa olsun ifadelerden oluşan işlem bloğu en az bir sefer çalışır. Eğer koşul sağlanamıyorsa sonsuz döngüye girilebilir. Bu nedenle bu döngü kullanılırken dikkat edilmelidir.



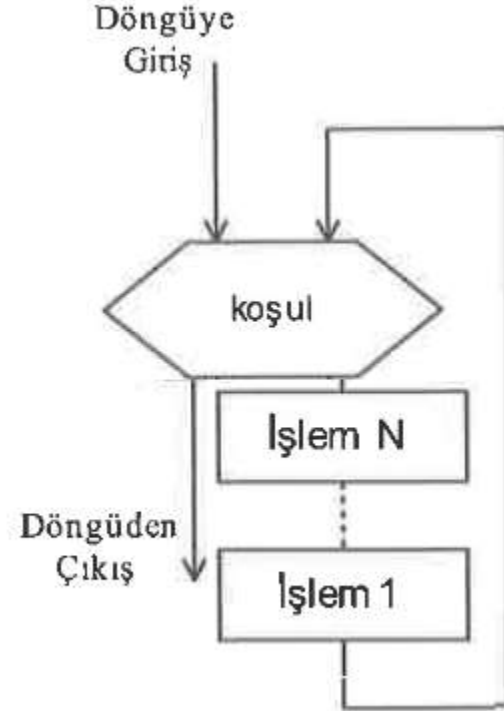
Döngü Çeşitleri

2. While Döngüleri

- While döngüsü de tekrarlı ifadeleri işlemek üzere geliştirilmiş bir başka döngüdür. Bu döngü yapısında koşul, döngünün ilk satırında yer alır. Koşul doğru olduğu sürece döngünün bloğunda yer alan komutlar işlenir.

While *Koşul* Do
İfade(i)
Repeat;

```
while (koşul) {  
    İşlem 1;  
    ...  
    İşlem N;  
}
```



Döngü Çeşitleri

3. for Döngüleri

- Kaç defa döneceği önceden bilinen durumlarda kullanılan bir döngü türüdür.
- Bu döngüde dönme adedini tutan bir döngü sayacı (örnekte x) değişkeni kullanılır. Bu değişken başlangıç olarak verilen değerden başlar, eşit değerli artımlarla veya azalışlarla test koşulu TRUE olduğu sürece devam eder. Bu esnada döngü bloğunda bulunan tüm ifadeler çalıştırılır.

C - LANGUAGE

declare variable
(optional) initialize test increment or
decrement

```
for(int x = 0; x < 100; x++){  
    println(x);  
}
```

BASIC -LANGUAGE

```
FOR i = 10 TO 1 STEP -1  
NEXT
```

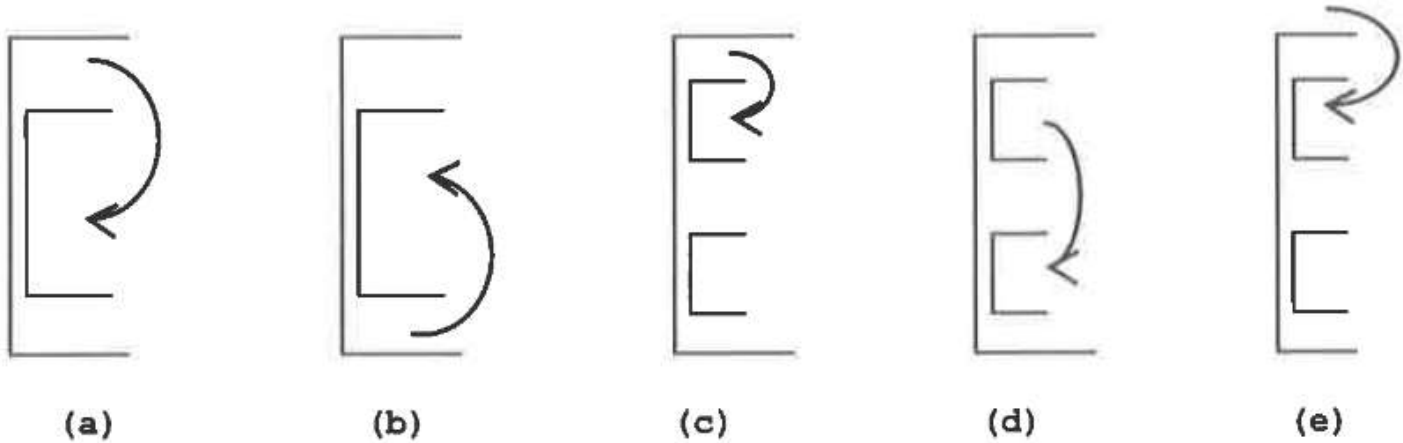
PYTHON - LANGUAGE

```
for i in range(1, 6):  
    print(i)
```

Döngülerde yanlış kullanım

Döngü Bloklarına Döngü Dışından Girilmesi

Döngü bloklarının içine dışarıdan *Goto* deyimi veya komutu ile girilemez. Her döngü kendi içerisinde bir bütündür. Aşağıdaki şekilde geçersiz veya kurulmaması gereken döngü yapılarına örnekler verilmiştir:



Şekil 3.6. Döngülerde izin verilmeyen durumlar

Şekil 3.6 da döngü bloğu içine girilmek istenen çeşitli pozisyonlar görülmektedir. Bu işlemlerin tamamı kural dışı olup döngü içine kendi başlangıcı ile girilmesi gerekir.

Akış Diyagramları dışında tasarım yöntemleri

- Metinsel tanımlama
- N-S(Nassi-Schneiderman)şemaları
- W-O (Warnier-orr) Diyagramları
- Nesneye Yönelik Yaklaşım
(Gelecek dönem derste işlenecek)

Algoritma ve Akış Şeması

Alıştırma Soruları

1. 1 ile 10 arasındaki sayıların karelerinin toplamını ekranda gösterin
2. Girilen pozitif tamsayının kaç basamaklı olduğunu bulup ekranda gösterin
3. Girilen 20 tamsayıdan çift sayıların toplamının tek sayıların toplamına oranını ekranda gösterin
4. Girilen sayının 5'in kuvveti olup olmadığını ekranda gösterin