

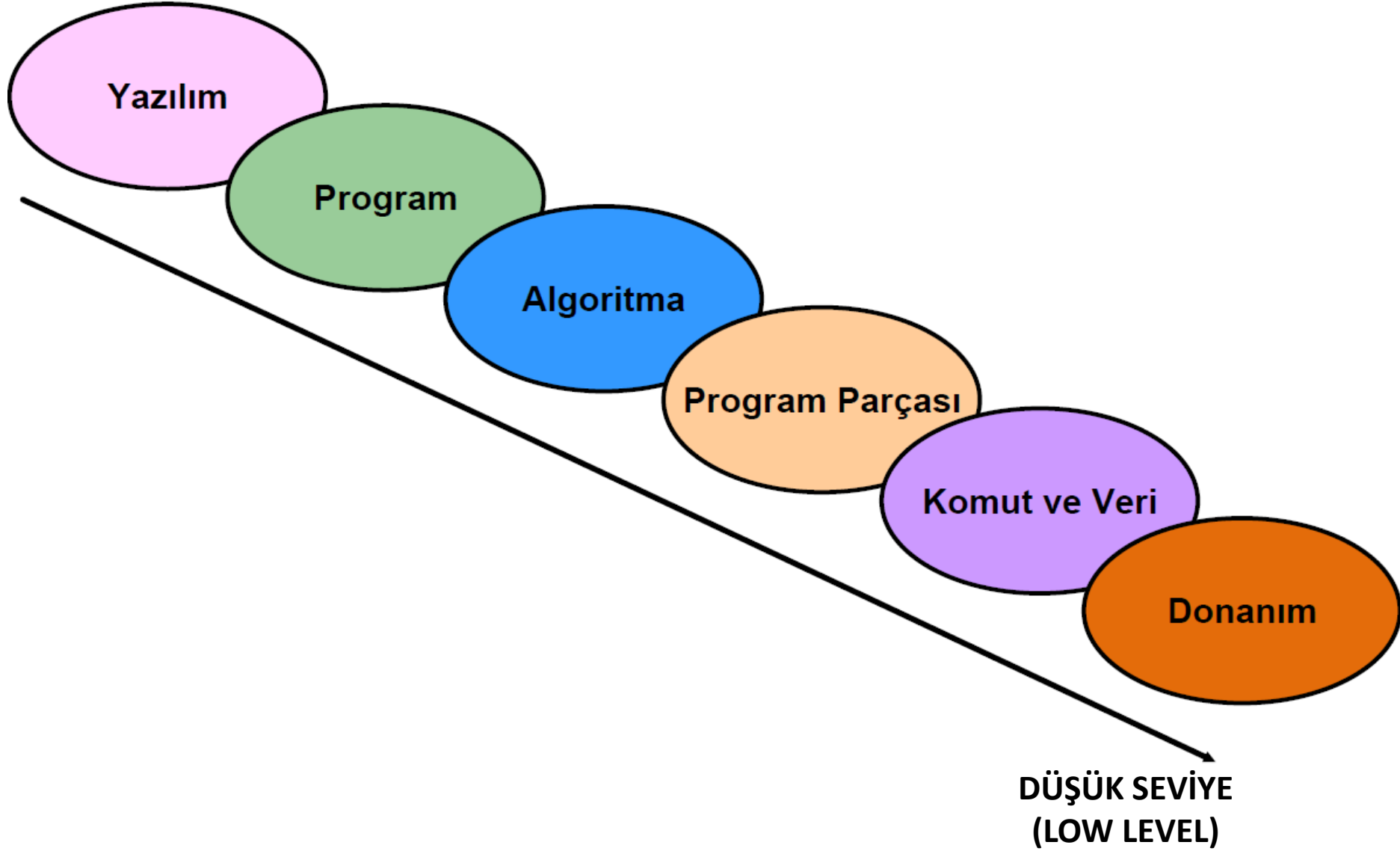


Algoritmik Program Tasarımı, Akış Şemaları ve Programlama

I) Algoritmik Program Tasarımı, Akış Şemaları

-  Algoritmik program tasarımı, verilen bir problemin bilgisayar ortamında çözülecek biçimde adım adım ortaya koyulması ve herhangi bir programlama aracıyla kodlanması sürecidir.
-  Akış şeması (flow chart), yapılacak bir işin veya programın şekilsel/grafiksel olarak ortaya koyulması veya başka bir deyişle tanımlanmasıdır.

**YÜKSEK SEVİYE
(HIGH LEVEL)**



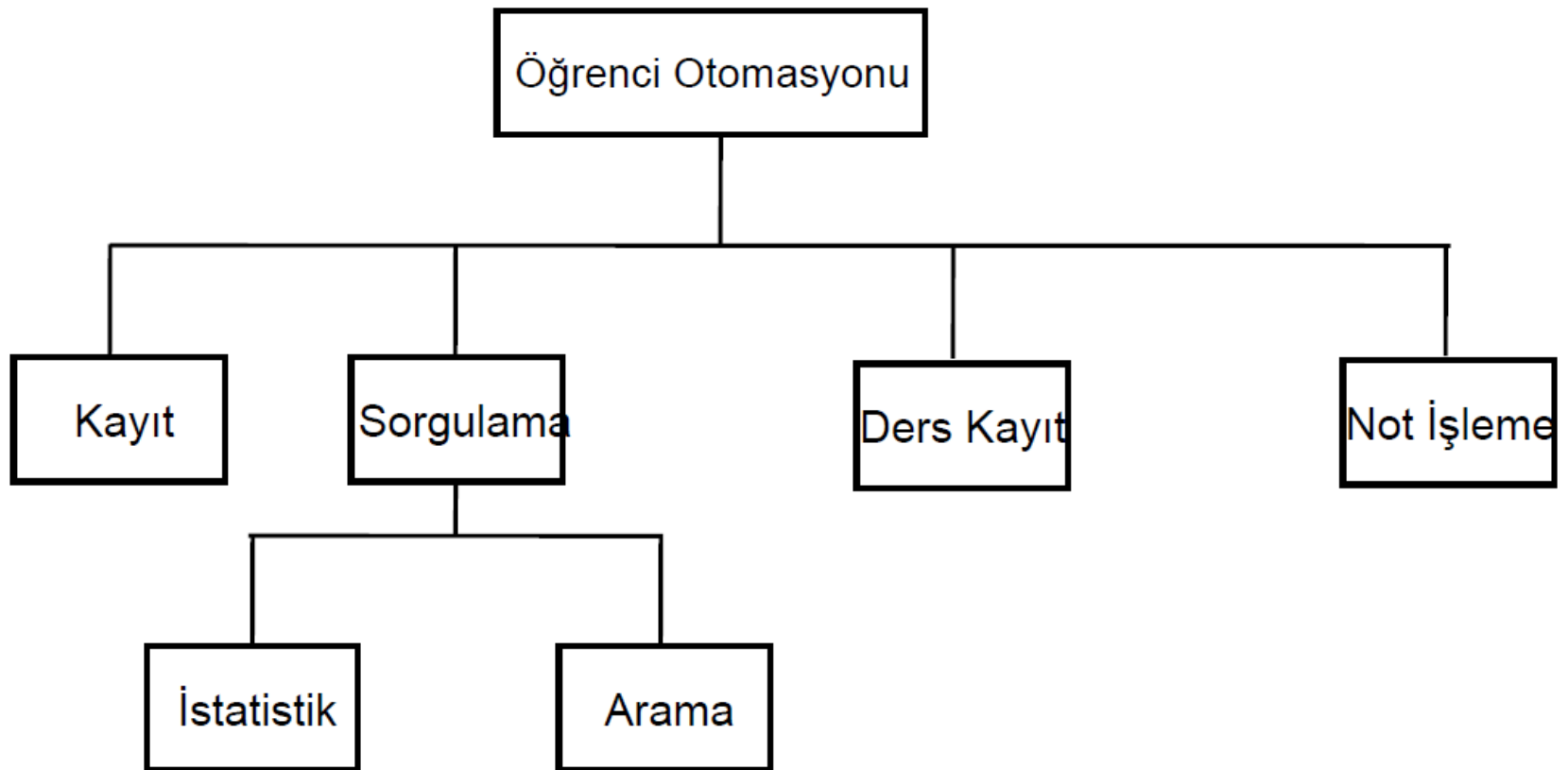
✓ Algoritma, genel olarak **tek bir problemi** çözecek davranışın, temel işleri yapan komutların/deyimlerin adım adım ortaya konulmasıdır.
Örn: Yalnızca sıralama veya arama yapan algortimalar.

✓ Tüm algoritmaları içerisinde barındırıp günlük veya iş yaşamındaki gereksinimi karşılayacak çözüm ise programdır. **Örn:** Küme elemanlarının okunması, onların sıralanması, diskte saklanması, istenilen bir elemanın aranması gibi işlemler bütünü programı oluşturur.

✓ Bir algoritmanın kısa bir parçacı program parçasıdır.



Yapısal şema (**The Structure Chart**), genel olarak yazılım veya programın ana parçalarını göstermek için kullanılır. **Örn:**



- ✓ Bir problem çözülürken biri **algoritmik**, diğeri **herustic** (sezgisel) olarak adlandırılan iki yaklaşım vardır.
- ✓ Algoritmik yaklaşımda, çözüm için olası yöntemlerden en uygun olanı seçilir ve yapılması gerekenler adım adım ortaya konulur.
- ✓ Herustic yaklaşımda ise, çözüm açıkça ortada değildir. Tasarımcının deneyimi, birikimi ve o andaki düşüncesine göre problemi çözecek birşeylerin şekillendirilmesiyle yapılır. Program tasarımcısı, algoritmik yaklaşımla çözemediği, ancak çözmek zorunda olduğu problemler için bu yaklaşımı kullanır.

Kaba ve Gerçek Kod (Pseudo Code ve Real Code)



- ✓ Pseudo Code (Kaba Kod), bir algoritmanın yarı programlama dili kuralı, yarı konuşma diline dönük olarak ortaya koyulması/tanımlanmasıdır. Bu şekilde gösterim algoritmayı genel hatlarıyla yansıtır.
- ✓ Real Code (Gerçek Kod), algoritmanın herhangi bir programlama diliyle, belirli bir veri yapısı üzerinde gerçekleştirilmiş halidir.

Bir algoritmanın gerçek kodu, yalnızca tasarlandığı veri yapısı üzerinde koşar; veri yapısı değiştirildiğinde algoritmanın gerçek kodu üzerinde oynamalar yapılmalıdır.

Pseudo Kod Örnekleri

- Pseudo kod örnekleri:
 - assign (ilkdeğer) value (100)
 - if (sayı<x)
 then (ilk satıra git)
 else (son satıra git)
 - while (döngü<girilen_değer)
 do assign (toplamlam) value (toplamlam+döngü)

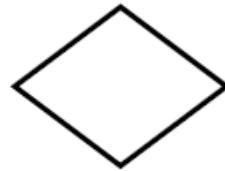
Akış Şemaları (Flow Charts)

-  Akış şeması, algoritmanın görsel/şekilsel olarak ortaya koyulmasıdır. Problemin çözümü için yapılması gerekenleri, başından sonuna kadar, geometrik şekillerden oluşan simgelerle gösterir.
-  Her simge, yapılacak bir işi veya komutu gösterir.

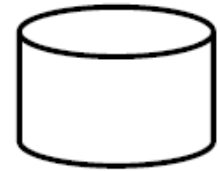
Akış Şemaları için Kullanılan Bazı Semboller



Başlama / Bitiş



Karar Verme



Dosya İşlemi



İşlemler



Döngü



Çıkış (Kağıt...)




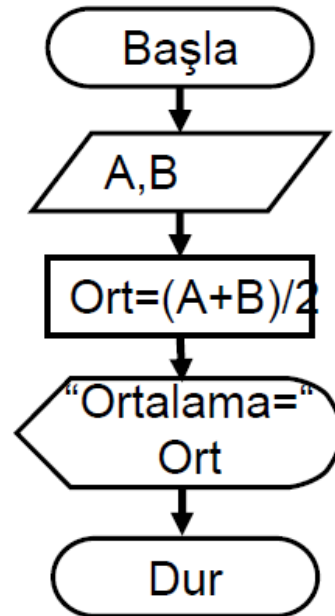
Giriş (Klavyeden
..)



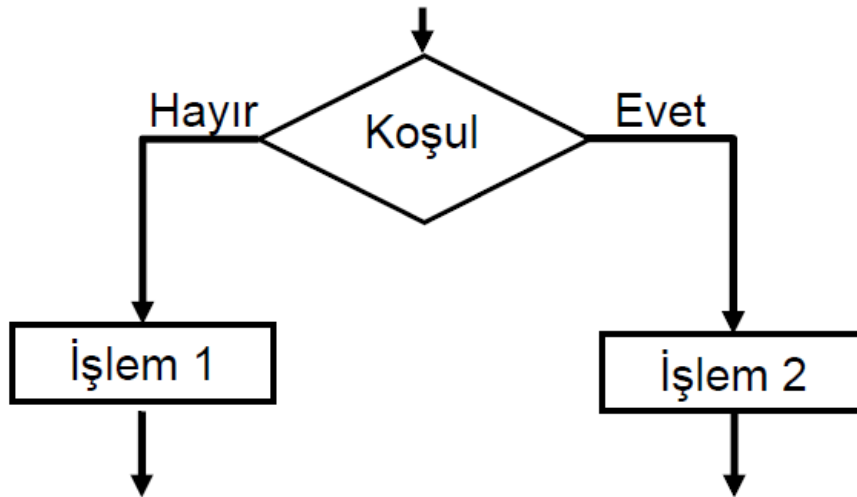
Çıkış
(Ekran...)

Örnek

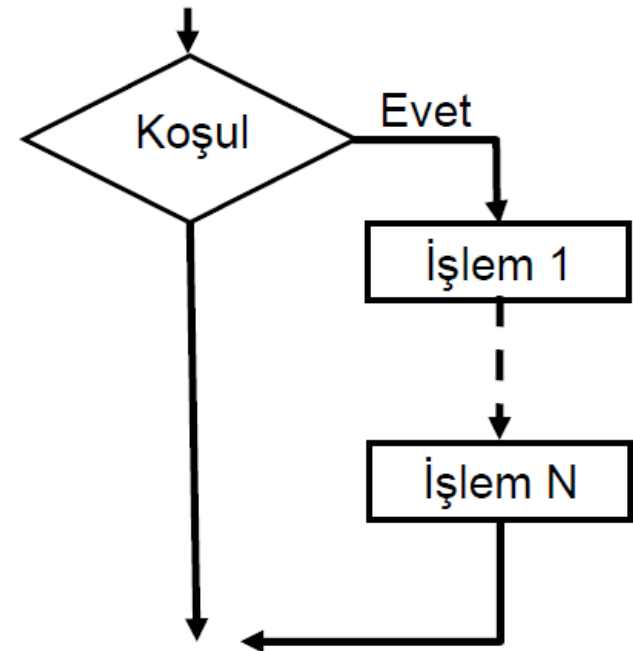
 Klavyeden girilen iki sayıyı alıp aritmetik ortalamasını hesaplayan ve sonucu ekrana yazan bir algoritmanın akış şeması?



Karar Verme



a) Koşulun durumuna bağlı olarak 2 farklı işlem vardır.

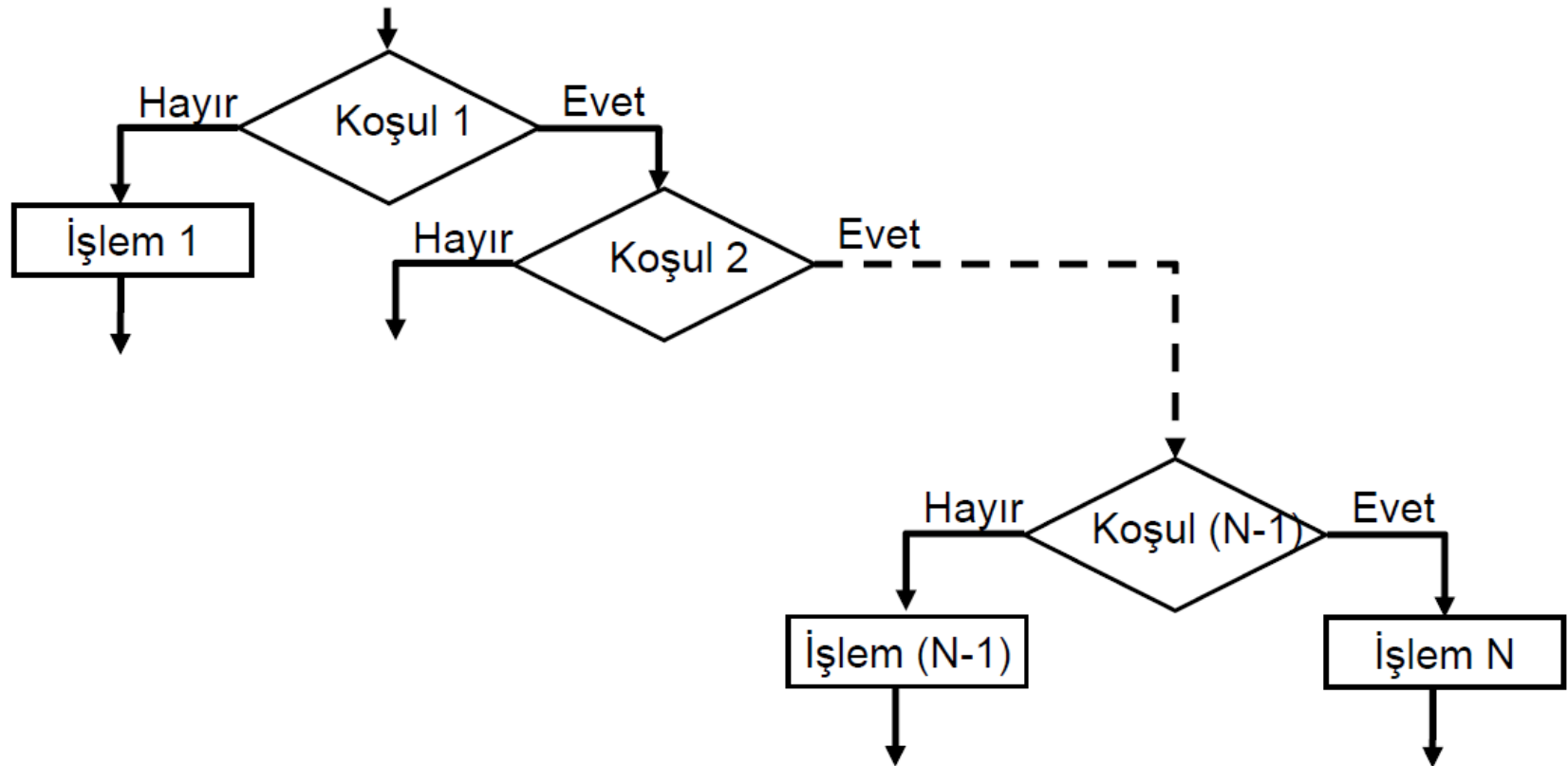


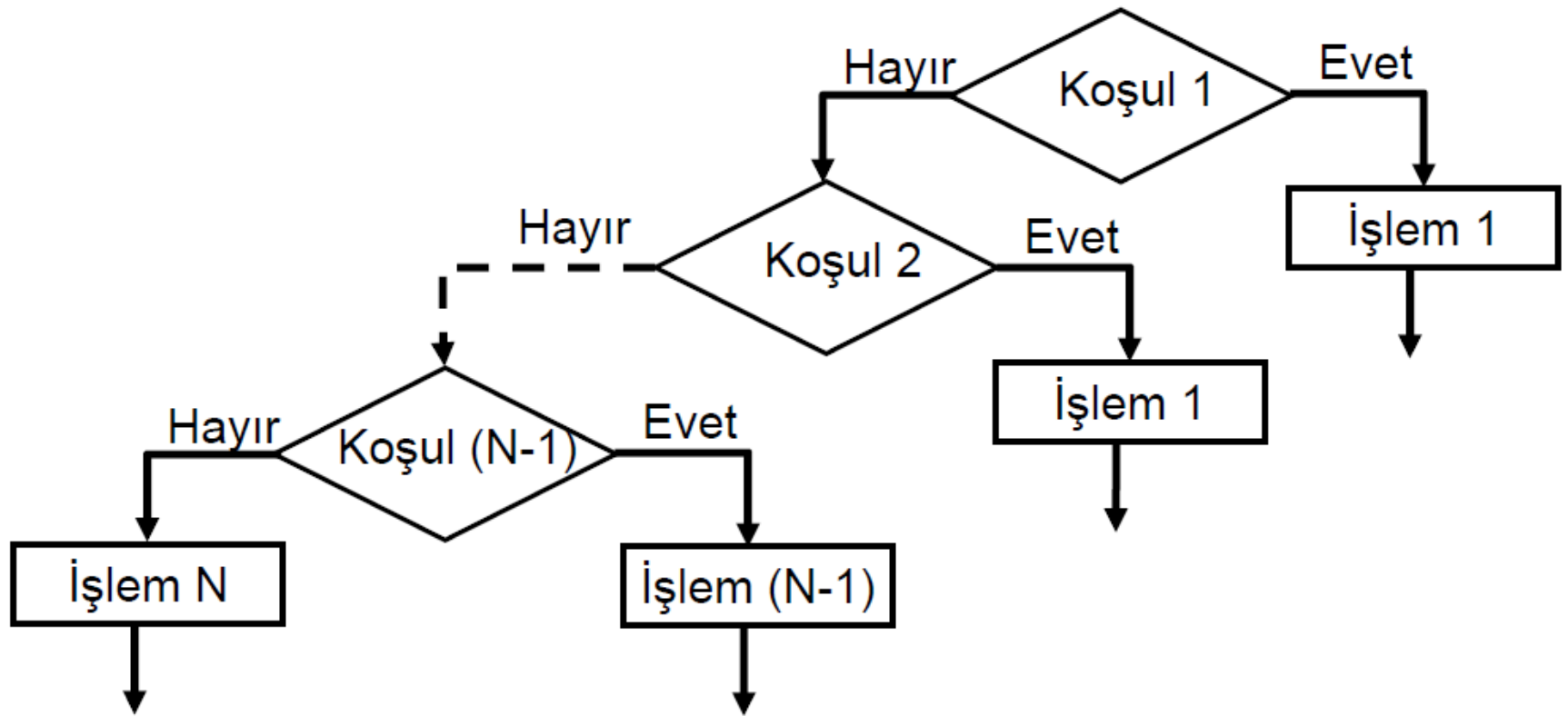
b) Olumsuz koşulda yapılacak işlem yoktur; olumlu olması durumunda ise N adet işlem yapılacaktır.

Karar Verme (Devam)



Bu yapıyı art arda birden çok kez kullanıp aşağıdaki gibi bir karşılaştırma dizisi oluşturulabilir.



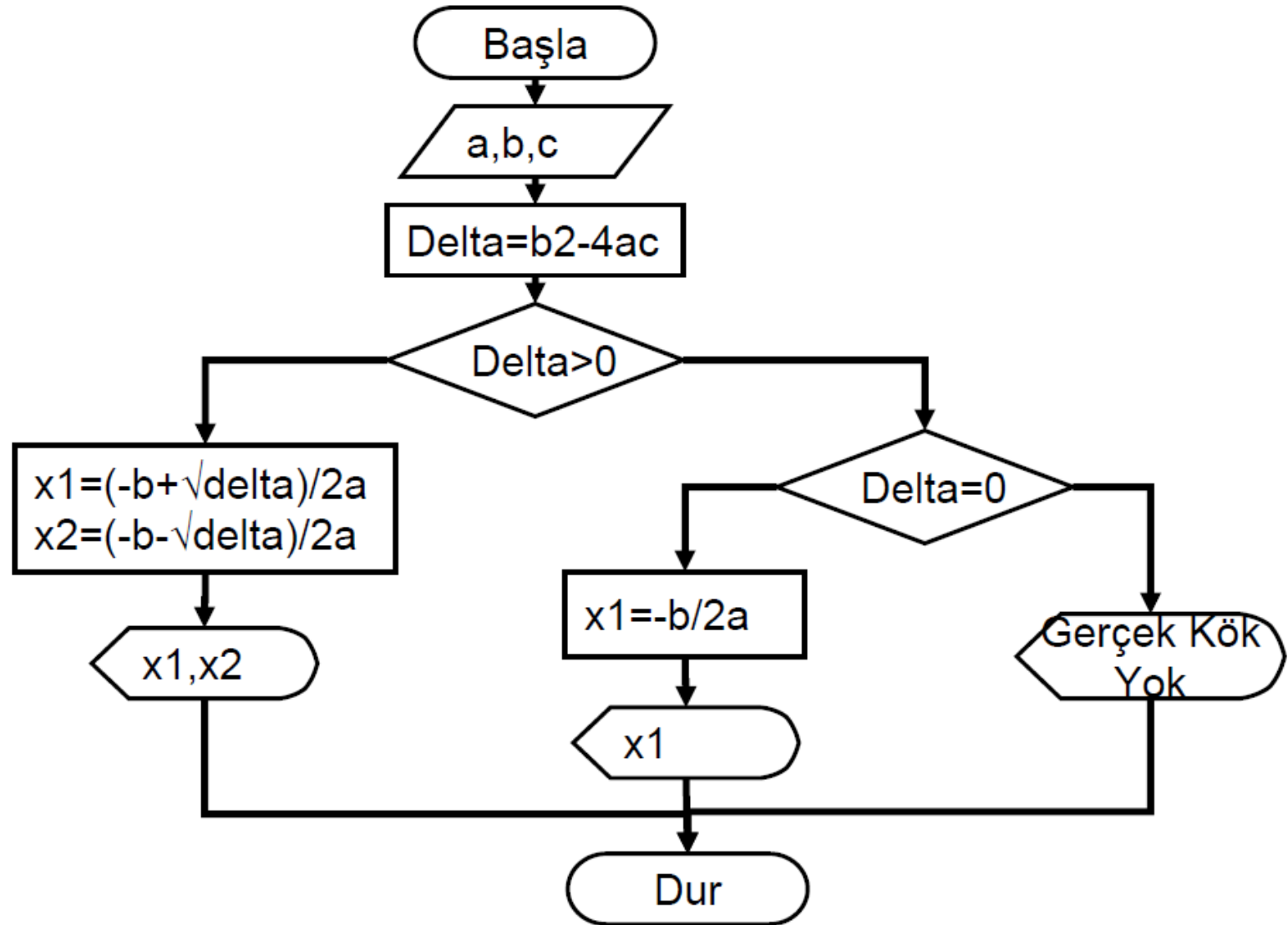


Olumsuz taraftan içiçe karşılaştırma simgesi örneği

Örnek



$ax^2 + bx + c = 0$ şeklindeki ikinci dereceden bir denklemin köklerini bulan algoritmayı tasarlayıp akış şeması ile gösteriniz.



Döngü Yapısı



Bu yapı kullanılırken, döngü sayacı, koşul bilgisi ve sayacın artım bilgisi verilmelidir. Döngü sayacı kullanılmıyorsa sadece döngüye devam edebilmek için gerekli olan koşul bilgisi verilmelidir.



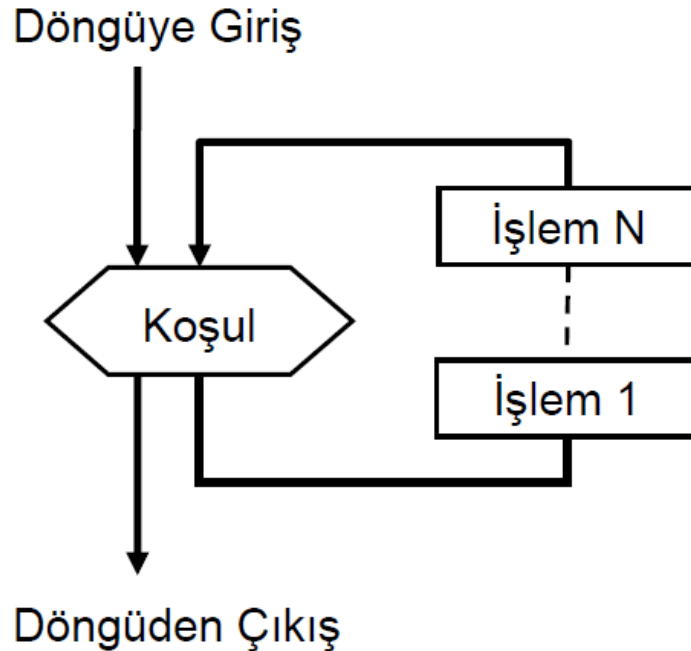
Genel olarak çoğu programlama dilinin döngü deyimleri ;

- While
- Do- while
- For

gibi yapılar üzerine kurulmuştur. Farklı dillerde bu yapılara farklı alternatifler olsa da döngülerin çalışma mantığı genel olarak benzerdir.

While Deyimi

- ✓ Koşul daha çevrim içerisine girmeden sınanır. Koşul olumsuz olduğunda çerime hiç girilmez ve döngü içerisinde yapılması gerekenler atlanır.



- ✓ Çevrim içerisinde koşulu etkileyen kod olmalıdır.

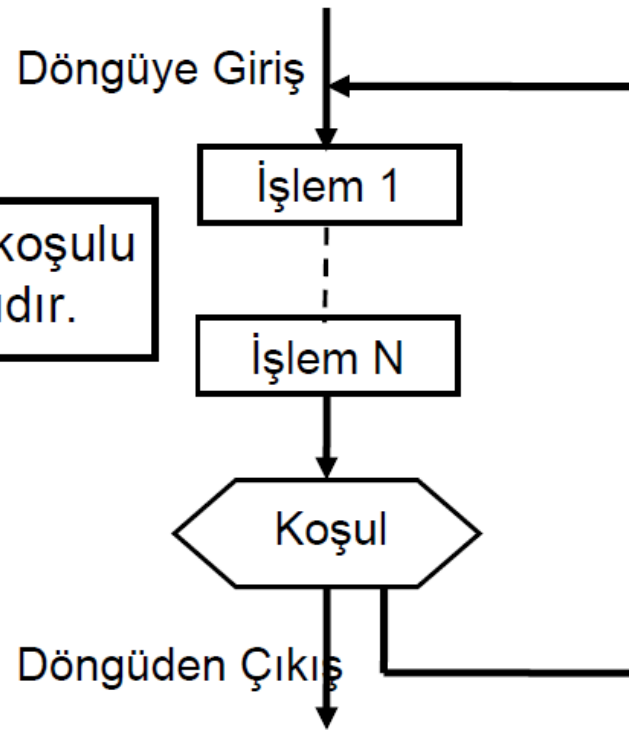
Do-While Deyimi



Bu döngü deyiminde, çevrim en az bir defa olmak üzere gerçekleşir. Çünkü koşul sınaması döngü sonunda yapılmaktadır. Eğer koşul sonucu olumsuz ise bir sonraki çevrime geçilmeden döngüden çıkılır. Çevrimin devam edebilmesi için her döngü sonunda yapılan koşul testinin olumlu sonuçlanması gerekir.

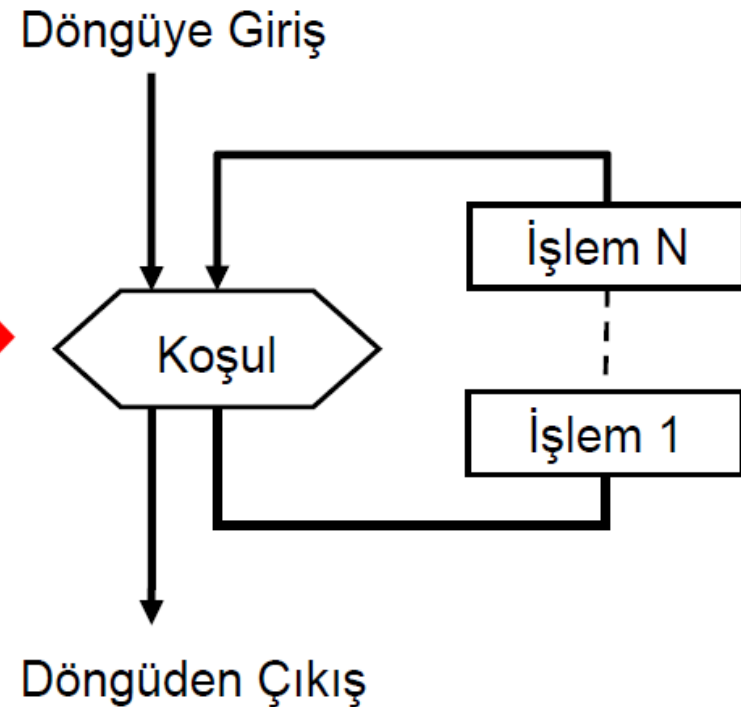
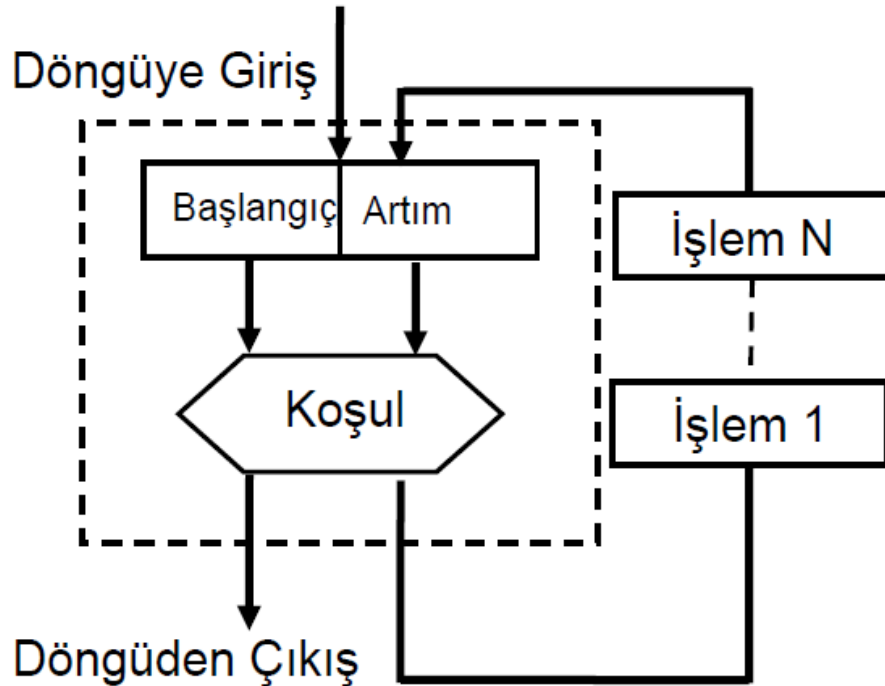


Çevrim içerisinde koşulu etkileyen kod olmalıdır.



For Deyimi

- ✓ Diğer deyimlerden farklı olarak, döngü sayacı doğrudan koşul parametreleri düzeyinde verilir.
- ✓ Döngü girmeden önce sayaç değişkenine başlangıç değeri atanmakta ve daha sonra koşula bakılmaktadır. Döngü içerisinde belirtilen işlemler yapıldıktan sonra sayaç değişkeni arttırılmaktadır.

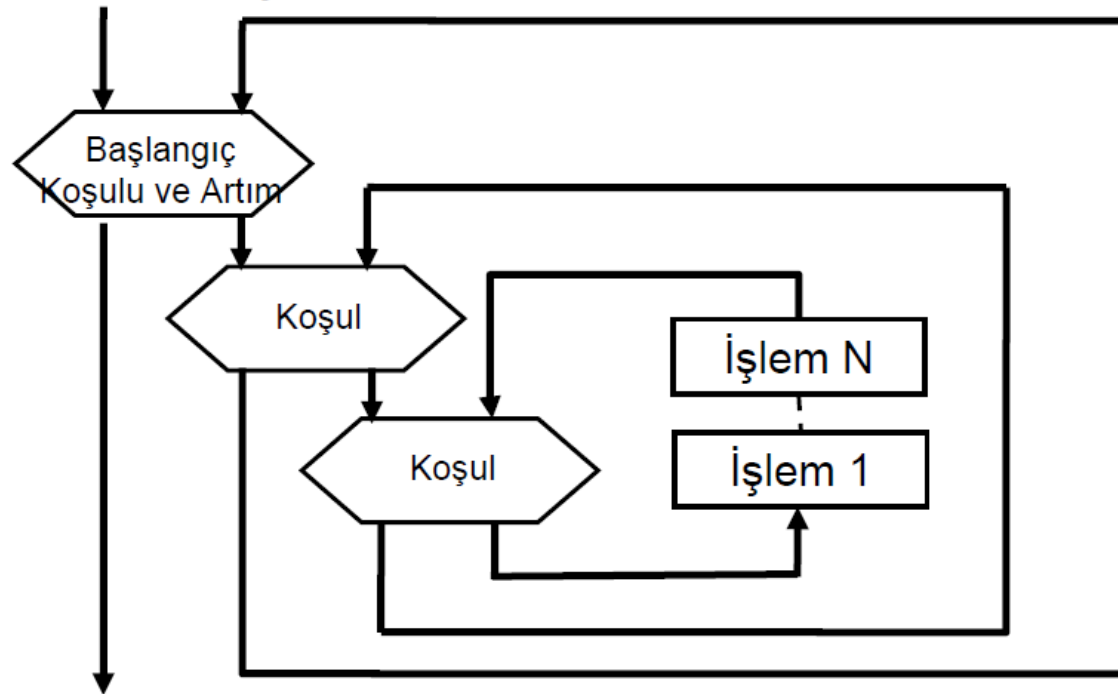


İççe Döngülerin Kullanılması



İççe döngü kurulurken en önemli unsur, içteki döngü sonlanmadan bir dıştaki döngüye geçilmemesidir. Diğer bir deyişle döngüler birbirlerini kesmemelidir.

Döngüye Giriş



Döngüden Çıkış



En içteki döngü bir dıştaki döngünün her adımında N kez tekrarlanır.

Örnek (Faktöriyel Hesabı)



$$n! = 1.2. (n-1).n$$

şeklinde ifade edilir.



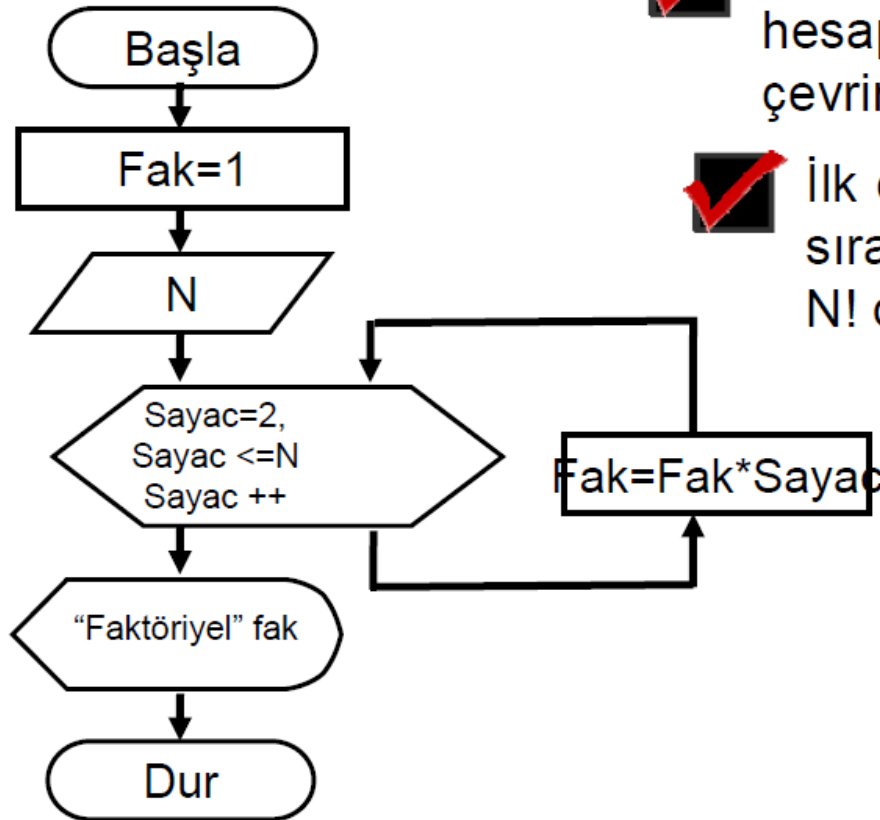
N ile hangi sayının faktöriyelini hesaplanacağı belirlenir ve N çevrimlik bir döngü kurulur.



İlk çevrimde 1!, ikinci çevrimde 2! ve sırayla N'inci çevrim sonucunda da N! değeri hesaplanmış olur.



Sayac > N koşulu oluştuğunda döngüden çıkılır ve elde edilen sonuç dış ortama aktarılır.



Örnek

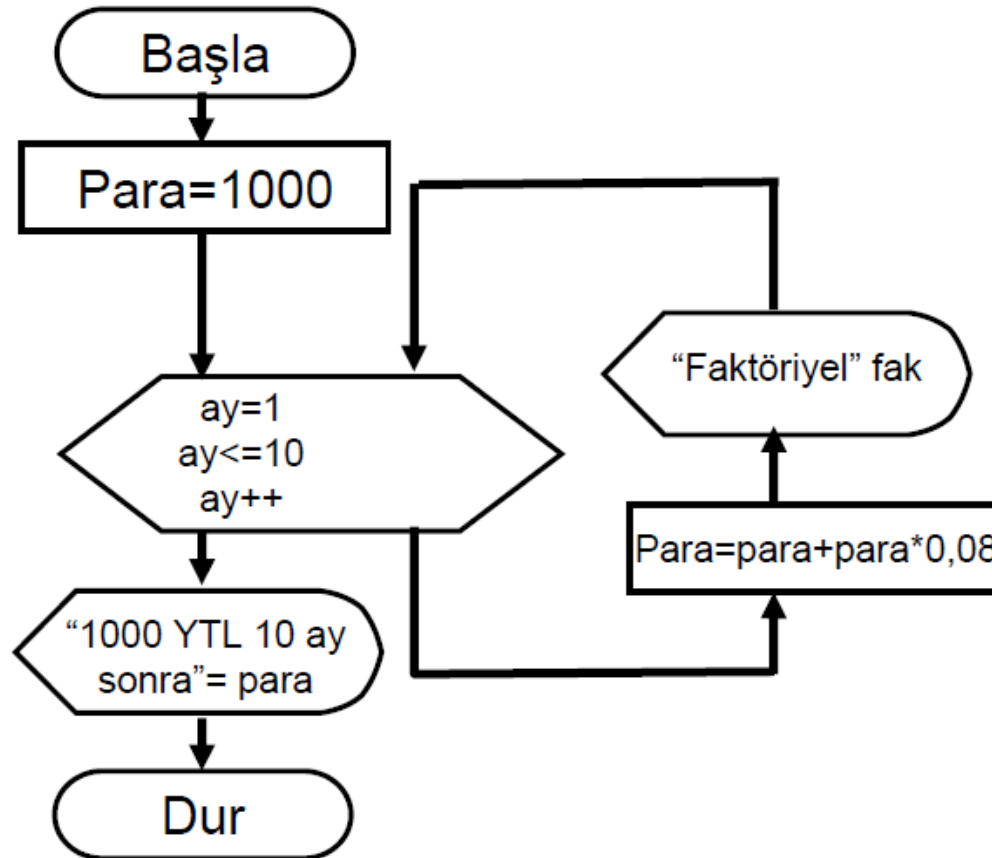


Bankaya aylık getirisi %8 ile 1000 YTL para yatırılmıştır. Buna göre;

a)10 ay sonrası için paranın ne kadar olacağını ve bu süre içerisinde her ay sonunda ne kadar olacağını hesaplayıp ekrana yazan programın akış şemasını nasıldır?

b)Paranın kaç ay sonra 5000 YTL olacağını hesaplayan programın akış şeması nasıldır?

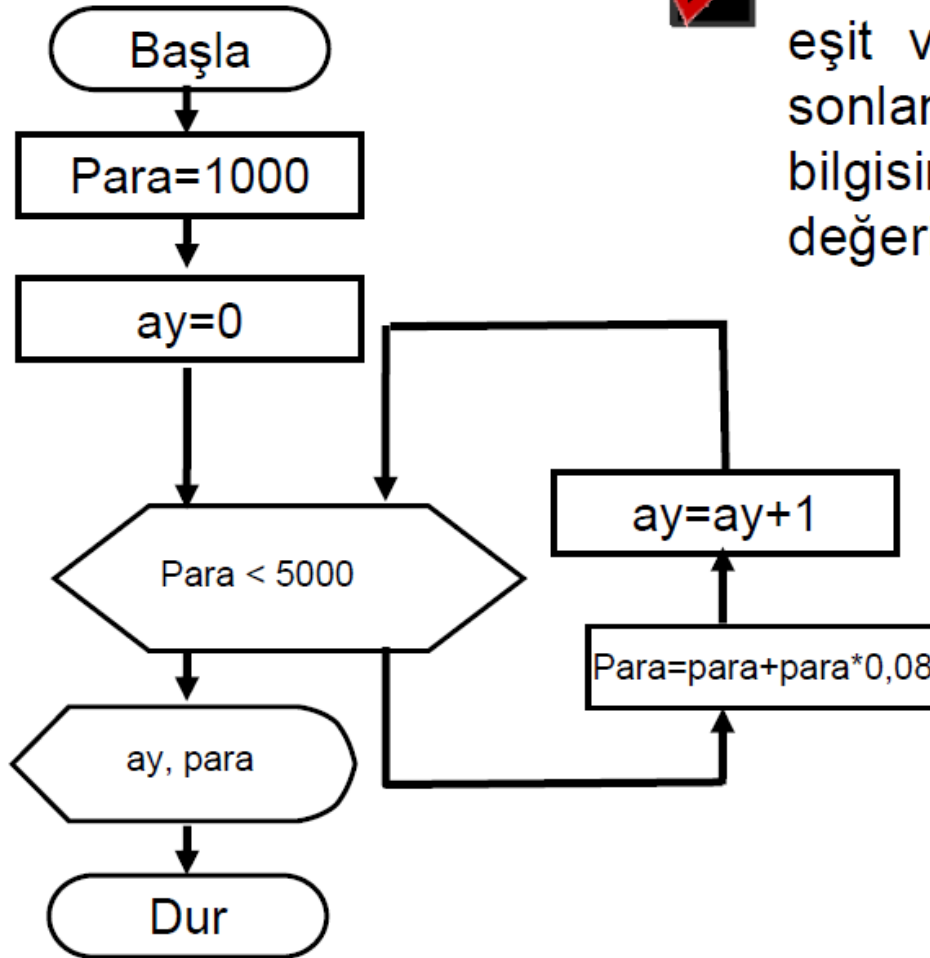
a)



b)



Ne zaman, para değeri 5000'den eşit veya büyük olduğunda döngü sonlanır ve döngü içerisindeki ay bilgisinin son hali bize istediğimiz değeri verir.



II) Programlama

- ✓ Çözülmesi söz konusu olan bir problem çözümlenip, algoritması ortaya konulduktan sonra sıra bilgisayar programının hazırlanmasına gelir.
- ✓ **Programlama**, bir işlemin bilgisayarın anlayabileceği bir biçime dönüştürülmesi işlemidir.



Bilgisayarların asıl işlevleri, büyük miktarda verileri işlemektir. Bu yüzden bilgisayar verileri okuyabilmeli, saklayabilmeli, aralarından gerekeni seçebilmeli ve onların üzerinde gereken işlemleri yapabilmelidir.



Bütün bu işlemleri, bilgisayara neyi nasıl hangi sırayla yapması gerektiğini gösteren programlar aracılığıyla gerçekleştirir.



Programlama işlemi ise programlama dilleri aracılığı ile yerine getirilir.

Programlama Dilleri



Farklı problem alanlarına göre farklı dillerden söz etmek mümkündür. Dilin seçimi bu yüzden önemli olabilir. Probleme uygun dilin seçilmesi programın geliştirilmesinde kolaylık ve hızlı çalışma sağlayabilir.

- ✓ Programlama dillerinin tümü simgeseldir. Yani bilgisayarın anlayacağı işlemleri insanların da anlayabileceği simgelerle ifade ederler.
- ✓ Örnek bir program içersinde yer alan “write” şeklindeki bir komutun yazma işlemini gerçekleştirdiğini söylemek oldukça kolaydır.
- ✓ Bu gibi komut ve ifadeler özel programlar aracılığı ile bilgisayarın anlayabileceği dile çevirirler. Bu programlara derleyici ya da yorumlayıcı adı verilir.

- ✓ Programın, bir programlama diline bağlı olarak hazırlanan simgesel kodlardan oluşan kısmına kaynak program (source code) adı verilir.
- ✓ Bu kaynak programın derlenip bilgisayarda çalışabilecek makine kodlarına çevrilmiş koduna ise nesne kodu (object code) ismi verilir.



Programlama dillerini seviyelerine göre dört kısımda sınıflandırmak mümkündür:

- Makine Dilleri
- Assembly Dilleri
- Yüksek Düzey Diller
- Dördüncü Kuşak Diller

Makine Dilleri



Doğrudan işlemcinin komut setinde yer alan makine kodlarından oluşur. En düşük seviyeli dil olarak kabul edilir. Komutlar ve veriler tamamen ikili düzende ifade edilir.

Assembly Dilleri

- ✓ Makine dilinde yer alan kodların anımsatıcı sembollerle (mnemonic symbols) ifade edildiği daha üst düzey bir dildir. Adresler ikilik değil simgesel olarak ifade edilir. Bu programların makine diline çevrilmesini sağlayan programlara assembler adı verilir.

Yüksek Düzeyli Diller

✓ Programcıya kolaylık sağlamak üzere geliştirilmiş tüm diller bu gruba girer. Komutlar yapılacak işlemin ingilizce karşılığı anımsatıcı kodlardır. Bu diller bilgisayar donanımından bağımsız olarak çalışabilirler. Yüksek düzey dillerde hazırlanan kodlar ancak makine diline çevrilip çalıştırılabilirler.

Bu dillere örnek olarak BASIC, C, C++, COBOL, PASCAL verilebilir.

Dördüncü Kuşak Diller

- ✓ Yüksek seviyeli dillere göre daha esnek yapıya sahip olan dördüncü kuşak diller programcının yapması gereken bir çok kodu otomatik olarak üretebilen özelliklere sahiptirler. Bu dillere en iyi örnek sorgulama dilleridir. Sorgulama dilleri veritabanları üzerinde çok az kod ile karmaşık sorguların yapılabilmesine olanak sağlayan araçlardır.

