

Algoritmalar

Kısaca algoritma belirli bir görevi yerine getiren sonlu sayıdaki işlemler dizisidir. Geniş anlamda ise algoritma, verilen herhangi bir sorunun çözümüne ulaşmak için uygulanması gerekli adımların hiç bir yoruma yer vermeksizin açık, düzenli ve sıralı bir şekilde söz ve yazı ile ifadesidir. Algoritmayı oluşturan adımlar özellikle basit ve açık olarak sıralandırılmalıdır.

Her algoritma aşağıdaki kriterleri sağlamalıdır.

1. Girdi: Sıfır veya daha fazla değer dışarıdan verilmeli.
2. Çıktı: En azından bir değer üretilmeli.
3. Açıklık: Her işlem (komut) açık olmalı ve farklı anlamlar içermemeli.
4. Sonluluk: Her türlü olasılık için algoritma sonlu adımda bitmeli.
5. Etkinlik: Her komut kişinin kalem ve kağıt ile yürütebileceği kadar basit olmalıdır.

PROGRAM YAZMA SÜRECİ

- Problemin farkına varmak,
- Problemi analiz etmek,
- Çözüm yolları düşünmek,
- İyi çözüm yolları seçip algoritma oluşturmak,
- Akış diyagramı çizmek,
- Uygun bir dilde kodlamak,
- Programı test etmek,
- Programı dağıtmak.

Program : Belirli bir problemi çözmek için bir bilgisayar dili kullanılarak yazılmış deyimler dizisi.

Bir problemi bilgisayar ile çözmek için geliştireceğimiz programın yazımında izleyeceğimiz adımlar:

- i) Problemin ne olduğunu kavra. Çözüm için gereksinimleri belirle.
- ii) Problemin girdilerini, çıktılarını ve diğer kısıtlama ve gereksinimleri belirle (bilgilerin giriş ve çıkış biçimlerinin nasıl olacağına kadar).
- iii) Problemin çözümünü veren algoritmayı yaz.
- iv) Algoritmayı bir programla dili ile yaz.
- v) Programın doğru çalışıp çalışmadığını test et. Bu testi değişik veriler (girdiler) için tekrarla.

Algoritmik çözüm yöntemlerine ilk örneği günlük yaşantımızdan verelim.

Örnek 1: Örneğimiz bir insanın evden çıkıp işe giderken izleyeceği yolu ve işyerine girişinde ilk yapacaklarını adım adım tanımlamaktadır.

Çözüm 1:

- Evden dışarıya çık
- Otobüs durağına yürü
- Durakta gideceğin yöndeki otobüsü bekle
- Otobüsün geldiğinde otobüse bin
- Biletini bilet kumbarasına at
- İneceğin yere yakınlaştığında arkaya yürü
- İneceğini belirten ikaz lambasına bas
- Otobüs durunca in
- İşyerine doğru yürü
- İş yeri giriş kapısından içeriye gir
- Mesai arkadaşlarıyla selamlaş
- İş giysini giy
- İşini yapmaya başla

Yukarıdaki örnekte görüldüğü gibi, evden işe gidişte yapılabilecek işlemler adım adım sırasıyla, kısa ve açık olarak tanımlanmaya çalışılmıştır. Yukarıdaki algoritma kişinin otobüsü kaçırmaya olasılığı düşünülmeden oluşturulmuştur. Kişi durağa geldiğinde bineceği otobüsü kaçırmış ise algoritmamız aşağıdaki şekilde değiştirilebilir.

Çözüm 2:

- Evden dışarıya çık Otobüs durağına yürü ,
- Otobüsün saati geçmiş?
- Durakta gideceğin yöndeki bir sonraki otobüsü bekle
- Bir sonraki otobüs gelene kadar 4. adımı uygula
- Otobüsün geldiğinde otobüse bin
- Biletini bilet kumbarasına at
- İneceğin yere yakınlaştığında arkaya yürü
- İneceğini belirten ikaz lambasına bas
- Otobüs durunca in
- İşyerine doğru yürü
- İş yeri giriş kapısından içeriye gir

- Mesai arkadaşlarıyla selamlaş
- İş giysisini giy
- İşini yapmaya başla.

Her iki örnekte görüldüğü gibi sorunu çözüme götürebilmek için gerekli olan adımlar sıralı ve açık bir biçimde belirlenmiştir. Algoritmanın herhangi bir adımındaki küçük bir yanlışlık doğru çözüme ulaşmayı engelleyebilir. Bu nedenle algoritma hazırlandıktan sonra dikkatle incelenmeli ve varsa adımlardaki yanlışlıklar düzeltilmelidir.

Programlamanın temeli olan algoritma hazırlanmasında dikkat çekici bir nokta, aynı sorunu çözmek için hazırlanabilecek olası algoritma sayısının birden çok olmasıdır. Başka deyişle, bir sorunun çözümü için birbirinden farklı birden fazla sayıda algoritma hazırlanabilir. Bu da gösteriyor ki herhangi bir problemin çözümü için birbirinden farklı yüzlerce bilgisayar programı yazılabilir.

Bir bilgisayar programı için hazırlanacak olan algoritma da aynı şekilde çözüm yolunu bilmeyen bir kişiye, çözüme ulaşmak için neler yapması gerektiği anlatılıyormuş gibi hazırlanmalı ve eksik bir nokta bırakmaksızın gerekli tüm adımları açık ve düzenli olarak içermelidir. Çözüm için kullanılacak bilgilerin nereden alınacağı, nerede saklanacağı ve çözümün program kullanıcısına nasıl ulaştırılacağı algoritma adımları arasında belirtilmelidir.

Aşağıda değişik işlemlere ilişkin algoritma örnekleri verilmiştir.

Örnek 2: İki sayıyı toplamak için gerekli programa ait algoritmanın oluşturulması.

Algoritma:

1. Birinci sayıyı gir
2. İkinci sayıyı gir
3. İki sayının toplamını yap
4. Toplamın değerini yaz
5. Bitir.

Bu tam bir algoritmadır. Sözcüklerin ortaya çıkaracağı yanlış anlamaların ortadan kaldırmak amacıyla semboller ve matematik dilini gerektiren bazı kısaltmalar kullanmak daha uygun olacaktır.

Bir algoritma yazılırken şu metod izlenmelidir:

- Programda kullanılacak elemanları temsil etmek üzere uygun isimler Veya değişkenler seç.
- Bazı isimlere başlangıç değeri olarak çözümün gerektirdiği uygun değerler ver.
- Gerekirse programa girilecek verileri düzenle.
- Cebirsel notasyon ve kararlar kullanarak aritmetik işlemleri gerçekleştir.
- Çıkışı düzenle.
- Bitir.

Yukarıda iki sayının toplanması için oluşturduğumuz algoritmayı bu yeni gereksinimlere uyarak yeniden yazalım.

Toplam adı için Z, Birinci sayı için X, İkinci sayı için Y değerleri kullanılırsa;

Algoritma:

1. X değerini gir
2. Y değerini gir
3. $Z \leftarrow X+Y$
4. Z' yi yaz
5. Bitir.

Görüldüğü üzere bu şekilde bir algoritma ile çözüm yolunu izlemek daha kolaydır. Bundan sonra verilen örneklerde bu tip algoritma kullanılacaktır.

Örnek 3: İki sayının ortalamasını bulan programa ait algoritmanın oluşturulması

Algoritma:

1. X değerini gir
2. Y değerini gir
3. $Z \leftarrow X+Y$
4. $Ort \leftarrow Z/2$
5. Ort değerini yaz
6. Bitir.

Bu örnekte Ort değeri ile iki sayının ortalaması temsil edilmiştir.

Örnek 4: Beş sayının toplamını ve ortalamasını veren programa ait algoritmanın oluşturulması

Toplam adı için Top

Ortalama adı için Ort

Girilen sayılar için X ,

Arttırma için Sayaç kullanılırsa

Algoritma:

1. Top ? 0, Sayaç ? 0
2. X'i gir
3. Top? Top+X
4. Sayaç ? Sayaç +1
5. Eğer Sayaç <5 ise A2'ye git
6. Ort? Top/5
7. Top ve Ort değerlerini yaz
8. Bitir.

Örnek 5: Kenar uzunlukları verilen dikdörtgenin alan hesabını yapan programa ait algoritmanın hazırlanması. Kenar uzunlukları negatif olarak girildiği durumda veri girişi tekrarlanacaktır.

Dikdörtgenin kısa kenarı : a

Dikdörtgenin uzun kenarı : b

Dikdörtgenin alanı: Alan

Algoritma:

1. a değerini gir
2. a<0 ise 1. adımı tekrarla
3. b değerini gir
4. b<0 ise 3. adımı tekrarla
5. Alan ? a*b
6. Alan değerini yaz
7. Bitir.

Örnek 6: Çapraz döviz kuru hesabi yapan programın algoritmasının oluşturulması. Bu algoritmanın oluşumunda veriler; 1 Amerikan dolarının TL karşılığı, hesaplanacak \$ miktarı, çıkış ise verilen \$'in TL karşılığı olacaktır.

Doların değeri : Doldeg

Girilen Dolar miktarı : Dolar

TL karşılığı : Tlkar

Algoritma:

1. Doldeg'i gir
2. Doldeg<0 ise 1. adımı tekrarla
3. Dolar'i gir

4. $Dolar < 0$ ise 3.adimi tekrarla
5. $Tlkar \neq Doldeg * Dolar$
6. $Tlkar$ değerini yaz
7. Bitir

Örnek 7: Verilen bir sayının faktöriyelini hesaplayan programın algoritmasının oluşturulması

sayının faktöriyeli :Fak

Faktöriyel değişkeni :X

Faktöriyeli hesaplanacak sayı :Y

Algoritma:

1. $Fak \neq 1, X \neq 0$
2. Y'i gir
3. $Y < 0$ ise 2. adimi tekrarla
4. $X \neq X+1$
5. $Fak \neq Fak * X$
6. $X < Y$ ise 4. adıma geri dön
7. Fak değerini yaz
8. Bitir.

Bu algoritmada 1. adımda X 'e 0 ve Fak değişkenine 1 değeri atanıyor. 2. adımda Y değeri giriliyor ve 3. adımda Y değerinin 0 dan küçük bir değer olup olmadığı denetlenerek, sonuca göre gerekli komut veriliyor. 4. adımda X'in değeri 1 arttırılıyor ve 5. adımda X için Fak değeri hesaplanıyor. 6. adımda X in değerinin faktöriyeli hesaplanacak sayıdan küçük olması durumunda 4. adımdan itibaren işlemlerin tekrarlanması komutu veriliyor, X' in değerinin Yiye eşit olması durumunda işlemler tamamlanarak hesaplanan değerin yazdırılması işleminden sonra programın çalışması sona ermektedir.

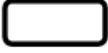
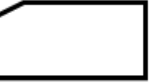


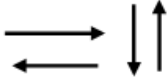

Akış Diyagramları

Geliştirilecek olan yazılımın genel yapısının şematik gösterimine akis diyagramı adi verilir. Akış diyagramları, yazılımı oluşturacak program parçalarını ve bu parçaların birbirleri ile olan ilişkilerini belirler. Bir bilgisayar programının oluşturulmasında akis diyagramlarının hazırlanması, algoritma oluşturma aşamasından sonra gelmektedir. Bilgisayar programının oluşturulması sırasında algoritma aşaması atlanarak, doğrudan akis diyagramlarının hazırlanmasına başlanabilir. Programlama tekniğinde önemli ölçüde yol almış kişiler bu aşamayı da atlayarak direkt olarak programın yazımına geçebilirler. Akış diyagramlarının algoritmadan farkı, adımların simgeler

sekinde kutular iwinde yazılmış olması ve adımlar arasındaki ilişkilerin (iş akışı) oklar ile gösterilmesidir.

Akış diyagramlarında kullanılan semboller, anlamları ve kullanım amaçları aşağıdaki tabloda verilmiştir.

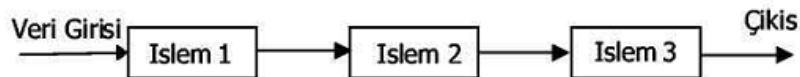
Tablo 1. Akış diyagramlarında kullanılan temel semboller ve anlamları

	Akış diyagramının başlangıç ve bitiş yerlerini gösterir. Başlangıç simgesinden çıkış oku vardır. Bitiş simgesinde giriş oku vardır.
	Dışardan veri girişi. Veri okutma işlemleri için.
	Aritmetik işlemler ve değışik atama işlemlerinin temsil edilmesi için kullanılır.
	Kontrol ve karar verme işlemlerini temsil eder.
	Oklar Diyagramın akis yönünü gösterir.
	Belgeye, yazıcıya, ekrana çıktı için kullanılır.

Bu semboller daha da çoğaltmak mümkün fakat temel olarak akış diyagramlarında bu şekiller kullanılacaktır.

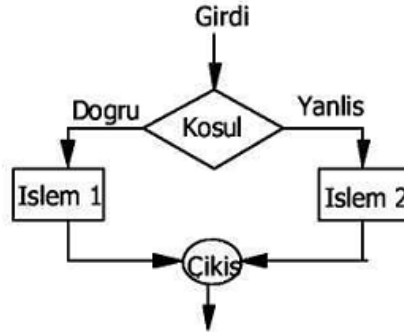
Ayrıntılı bir akış diyagramı, yazılımı oluşturan işlemleri ve ilişkilerini en küçük detayına kadar belirler.

Bir bilgisayar programının geliştirilmesinde kullanılan programlama dili ne olursa olsun bu programların akış diyagramlarında genel olarak yalnız üç basit mantıksal yapı kullanılır. Bu mantıksal yapılardan en basiti *sıralı yapıdır* (**Şekil 1.2**). Sıralı yapı, hazırlanacak programdaki her işlemin mantık sırasına göre nerede yer alması gerektiğini vurgular. Bu yapı sona erinceye kadar ikinci bir işlem başlayamaz.



Şekil 1.2 Sıralı Yapı

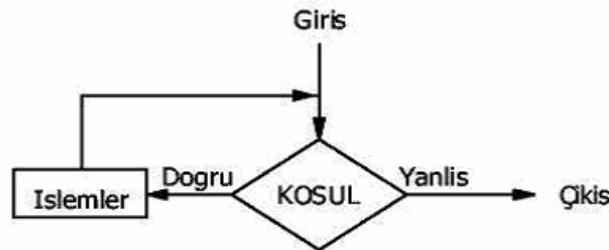
Mantıksal yapılardan ikincisi *Karar Verme* yapısıdır (**Şekil 1.3**). Programlama sırasında If...Then... Else yapısı ile tanıyacağımız bu mantıksal yapılar, birden fazla sıralı yapı seçeneğini kapsayan modüllerde, hangi şartlarda hangi sıralı yapının seçileceğini belirler.



Sekil 1.3 Karar Verme Yapisi

Üçüncü mantıksal yapı çeşidini *tekrarlı yapılar* (**Şekil 1.4**) oluşturmaktadır. Bu yapılara Pascal programlama dilinde *For* (**Şekil 1.4**), *While* ve *Repeat..Until* yapısı adı da verilir. Şartlara göre değişik işlem gruplarının yapılmasını sağlar. Bu yapı yukarıda sözü edilen iki yapının çeşitli kombinasyonların tekrarlanmasından oluşmuştur.

Söz konusu üç değişik yapı, değişik kombinasyonlarda kullanılarak istenilen işlevleri yerine getirecek programlar hazırlanabilir. Programların bu üç basit yapı ile sınırlandırılması program modüllerinin daha kolay tasarlanmasını sağlar.

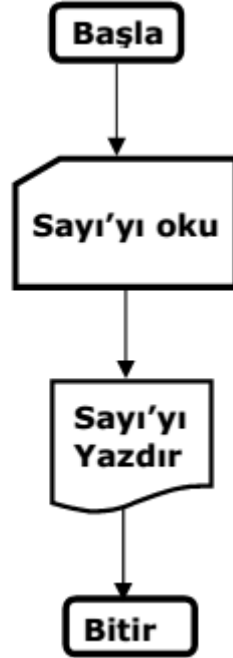


Şekil 1.4. Tekrarlı yapılar

Akış Diyagramı Örnekleri

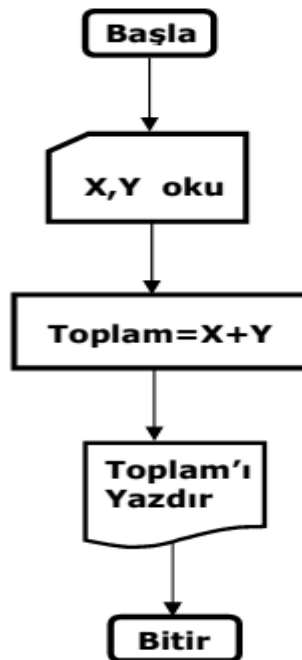
Örnek1

Dışardan girilen bir sayıyı okuyup bu sayıyı tekrar yazdıran programın akış şemasını çiziniz.



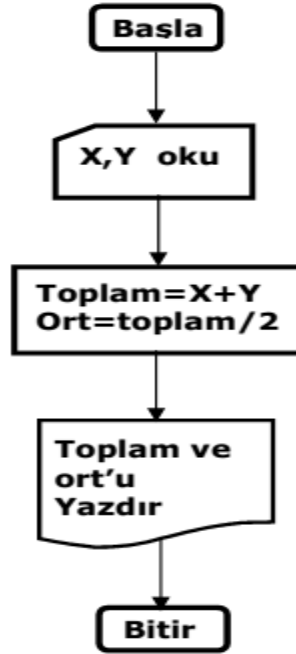
Örnek2:

Dışardan girilen iki sayıyı toplayıp sonucu ekrana yazdıran programın akış şemasını çiziniz.



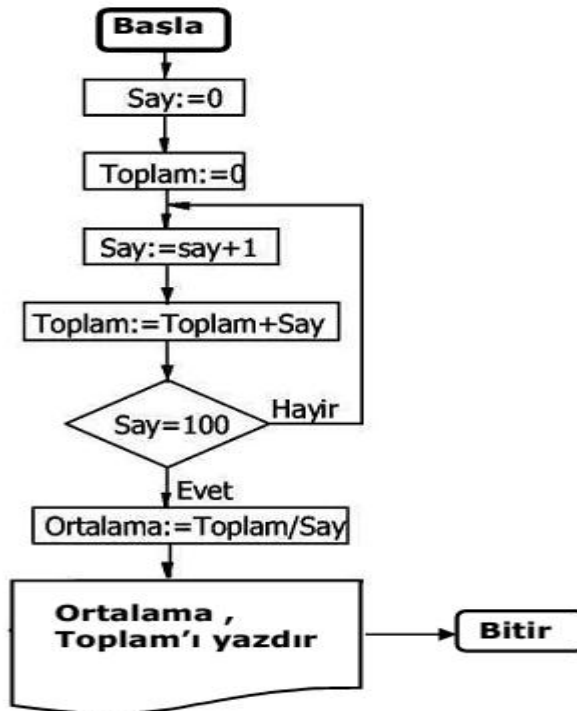
Örnek 3:

Dışardan girilen iki sayıyı toplayıp bunların ortalamasını bulup toplam ve ortalama sonucunu ekrana yazdıran programın akış şemasını çiziniz.



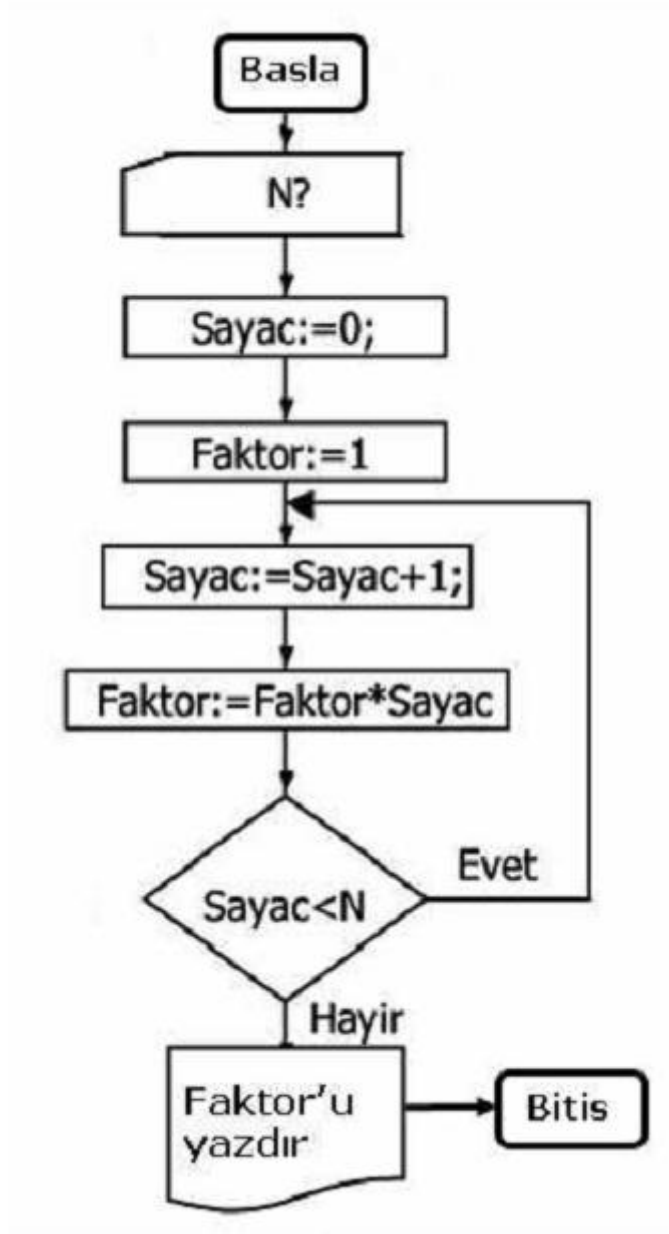
Örnek 4:

1'den 100'e kadar olan sayıların toplamalarını ve ortalamalarını veren programın akış diyagramını çiziniz.



Örnek 5:

Dışardan girilen N sayısının faktöriyelini hesaplayan programın akış diyagramını çiziniz.



Soru:

Bir sayıyı dışarıdan okuyup tekrar ekrana yazdıran problemin algoritmasını yazıp akış diyagramını çiziniz.

Algoritma	Akış Diyagramı
<ol style="list-style-type: none">1. Başla2. A sayısını oku3. Sayıyı yazdır4. Bitir.	<pre>graph TD; A[Başla] --> B[/Sayı'yı oku/]; B --> C[/Sayı'yı Yazdır/]; C --> D[Bitir];</pre>

Soru:

Dışarıdan bir sayı okuyup bu sayının faktöriyelini hesaplayan problemin algoritmasını yazıp akış diyagramını çiziniz.

Algoritma	Akış Diyagramı
<ol style="list-style-type: none"> 1. Başla 2. sayac=0, Faktör=1 al 3. Bir sayı oku (N) 4. sayac=sayac+1 al 5. faktör=faktör*sayac 6. Eğer sayac < N ise 4. adıma git 7. Faktör'ü yazdır 8. Bitir. 	<pre> graph TD Start([Başla]) --> Input[/N?/] Input --> Sayac0[Sayac:=0;] Sayac0 --> Faktör1[Faktör:=1] Faktör1 --> SayacInc[Sayac:=Sayac+1;] SayacInc --> FaktörMul[Faktör:=Faktör*Sayac] FaktörMul --> Decision{Sayac<N} Decision -- Evet --> SayacInc Decision -- Hayir --> Output[/Faktör'ü yazdır/] Output --> End([Bitir]) </pre>