Algoritma ve Akış Şemaları

- Algoritma Tanımı,
- Algoritma Özellikleri,
- Algoritma Tasarımı,
- Akış Şemaları,
- Dallanma Simgeleri
- Döngü Simgeleri,
- Akış Şeması Tasarımı,
- Akış Şeması Özellikleri,
- N-S Şeması,
- W-O Diyagramları

Muhammet Baykara
mbaykara@firat.edu.tr

Algoritma

- **Algoritma** bir problemin çözümünde izlenecek yol anlamına gelir. Tüm programlama dillerinin temeli algoritmaya dayanmaktadır.
- İki farklı **algoritma oluşturma** şekli vardır. Birincisi **akış diyagramı** şeklinde oluşan algoritmalar, ikincisi ise **düz yazı** (**pseudo kod**) ile oluşturulan algoritmalardır. **Pseudo kod** dediğimiz kısım ile algoritmayı akış diyagramı yerine basit kelimeler ile açıklayıcı bir şekilde sıralamaya dayanır. Kısacası yapılacak adımları tek tek yazmak olarak tanımlanabilir.

Algoritmada Olması Gereken Özellikler

- **Etkin ve Genel Olma:** Algoritma etkin olmalı ve gereksiz tekrarlar bulunmamalıdır. Gerektiğinde diğer algoritmalar içerisinde kullanılabilmelidir. Her koşulda ve giriş değerinde doğru sonuca ulaşmalıdır.
- **Sonlu Olma:** Her türlü olasılık için algoritma sonlu adımda bitmelidir. Algoritma sonsuz döngüye girmemelidir.
- **Yanılmazlık:** Algoritma tekrar yürütüldüğünde aynı giriş değerleri için aynı sonuç elde edilmelidir.

Algoritmada Olması Gereken Özellikler

- **Giriş Çıkış Tanımlı Olma:** Algoritmalarda giriş ve çıkış bilgileri olmalıdır. Dışarıdan gelen verilere giriş bilgisi denir. Bu veriler algoritmada işlenir ve çıkış bilgisini oluşturur. Çıkış bilgisi her algoritmada mutlaka vardır. Algoritmaların temel amacı giriş bilgisini işleyerek çıkış bilgisi oluşturmaktır.
- **Başarım:** Amaç donanım gereksinimi (bellek kullanımı gibi), çalışma süresi gibi performans kriterlerini dikkate alarak yüksek başarımlı programlar yazmak olmalıdır.

Örnek Bir Algoritma (Doğrusal Akış Şemaları)

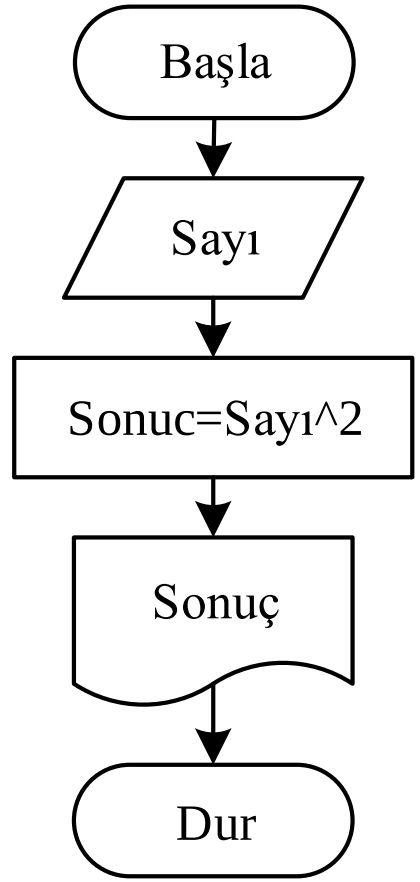
- En basit bir algoritma günlük doğal dil kullanılarak aşağıdaki gibi olabilir.

Algoritma 1:

1- Klavyeden girilen bir sayıyı oku.

2- Sayı üzerinde istenen işlemi yap; örneğin girilen sayının karesini hesapla.

3- Sonucu ekrana yaz.

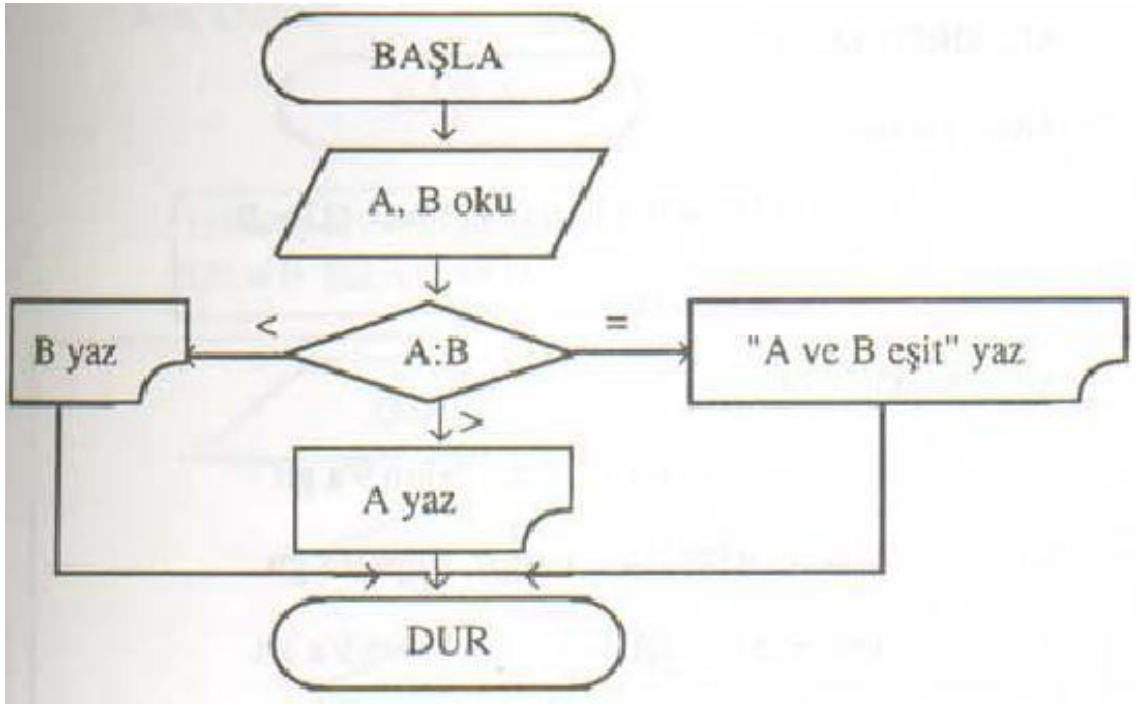


Örnek Bir Algoritma (MANTIKSAL AKIŞ ŞEMALARI)

- Geniş ölçüde mantıksal kararları içeren akış şemalarıdır. Hesap düzenleri genellikle basittir.

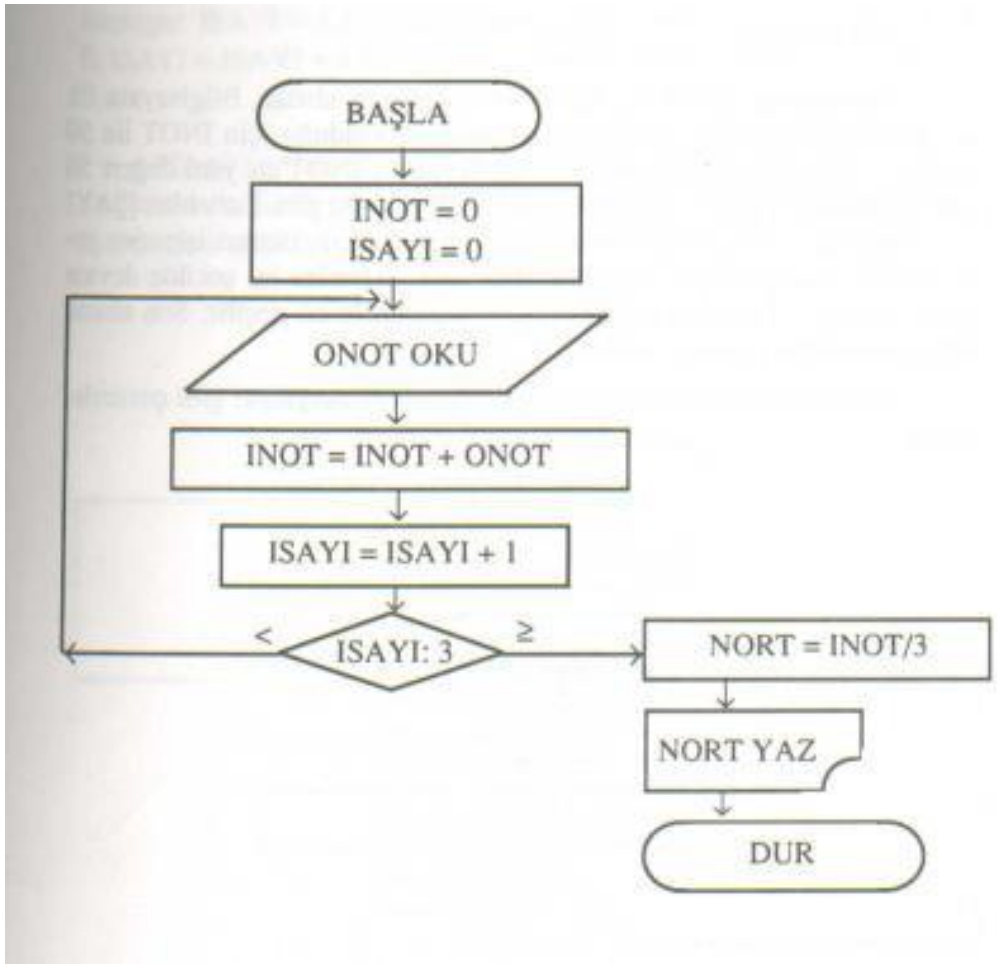
Algoritma 2:

- **Adım 1-**Başla
- **Adım 2-**A,B'yi oku
- **Adım 3-**A=B ise Adım 7'ye git
- **Adım 4-**A>B ise Adım 6'ya git
- **Adım 5-**B'yi yaz Adım 8'e git
- **Adım 6-**A'yı yaz Adım 8'e git
- **Adım 7-**"A ve B eşit' mesajını yaz
- **Adım 8-**DUR



Örnek Bir Algoritma (Yineli-Döngülü AKIŞ ŞEMALARI)

- Sorunun çözümü için, çözümde yer alan herhangi bir adım ya da aşamanın birden fazla kullanıldığı akış şemalarına denir.
- **Adım 1**-Başla
- **Adım 2**-INOT=0
- **Adım 3**-ISAYI=0
- **Adım 4**-ONOT oku
- **Adım 5**-INOT=INOT+ONOT
- **Adım 6**-ISAYI=ISAIY+1
- **Adım 7**-ISAYI<3ise Adım 4'e git
- **Adım 8**-NORT=INOT/3
- **Adım 9**-NORT YAZ
- **Adım 10**-DUR



Kaba (Pseudo) Kod ve Gerçek Kod

- **Kaba-kod,** bir algoritmanın yarı programlama dili kuralı, yarı konuşma diline dönük olarak ortaya koyulması/tanımlanmasıdır. Kaba-kod, çoğunlukla, bir veri yapısına dayandırılmadan algoritmayı genel olarak tasarlanır.
- **Gerçek kod** ise, algoritmanın herhangi bir programlama diliyle, belirli bir veri yapısı üzerinde gerçekleştirilmiş halidir.

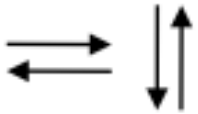
```
connect (server_ip, server_port);
while(connected)
{
  /* if previous time slot ends
     start a new time slot */
  if (current_time_slot)
     continue;
  /* when receive input from
     player, send command to server */
  if (player_input( ))
     send_command( );
  /*when receive update from server
    update game status */
  if (recv_update( ))
     world_update( );
  /*predict for all entities*/
  prediction( );
  /*render graphic, play audio,...etc*/
  render_effect( );
}
```

```
while(!game_end)
{
  /*if any player time_out, end game*/
  if (!time_out)   continue;

  while (recv_command( ))
  /*if players send a command, execute*/
  { execute_command( ); }
  /*AI determines NPC or item reaction*/
  perform_ai( );
  /*if player needs to be synchronized,
     sends update */
  for (n=0; n<player_number; n++)
  {
     if (update(n))
        send_update(n);
  }
}
```

client kodu | server kodu

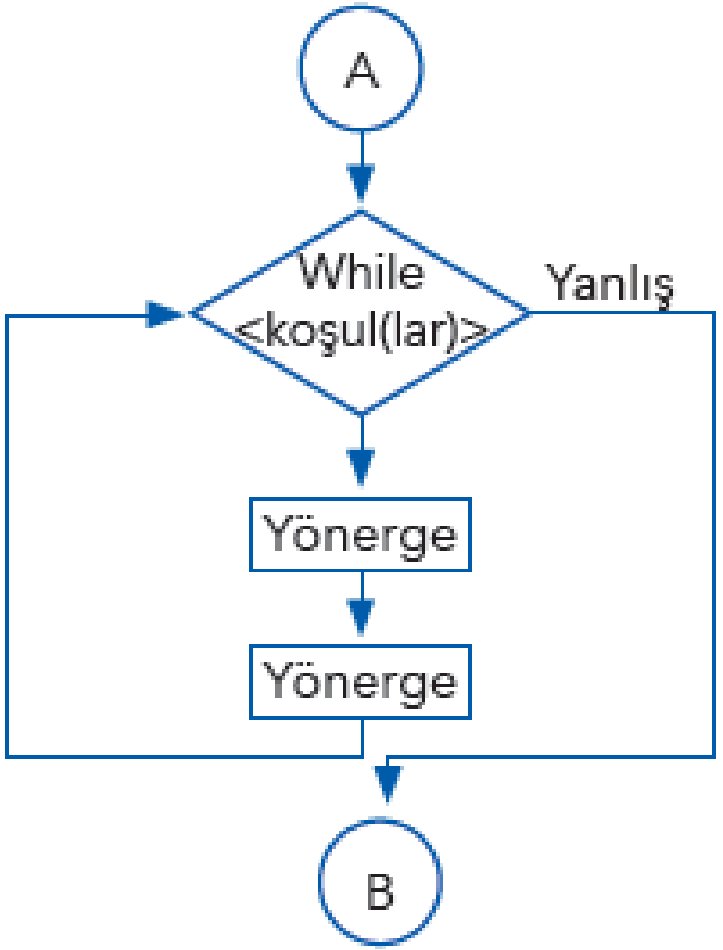
```php
public static function fromXml($xml)
{
    // Extract parameters from the XML response
    $elements = @$xml->xpath('/response/data');
    if (empty($elements[0]))
    {
        $message = "Couldn't extract details from the XML";
        throw new InvalidArgumentException($message);
    }

    $data = $elements[0];
    return new self(
        (float)   $data->amount,
        (string)  $data->currency,
        (string)  $data->id,
        (integer) $data->status,
        (string)  $data->status_msg
    );
}
```

Akış Şeması Simgeleri

Simge	İşlev	Simge	İşlev
	Başla/Bitir		Döngü
	Genel Girdi/Çıktı		Manyetik Disk
	Genel İşlem		Yordam Çağırma
	Denetim (Karar)		Akış Yönü
	Yazıcı Çıktısı		Bağlaç
	Görüntü Çıktısı		Sayfa Bağlacı
	Ele ile Girdi (Klavye)		

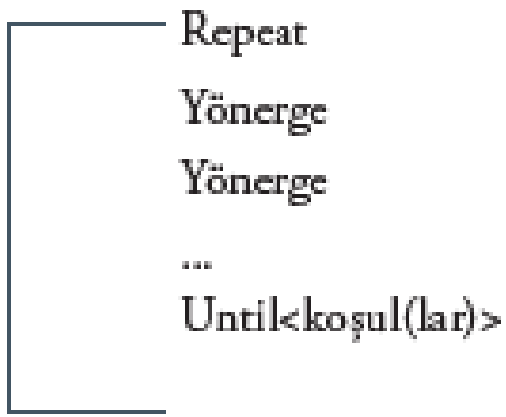
Döngüler

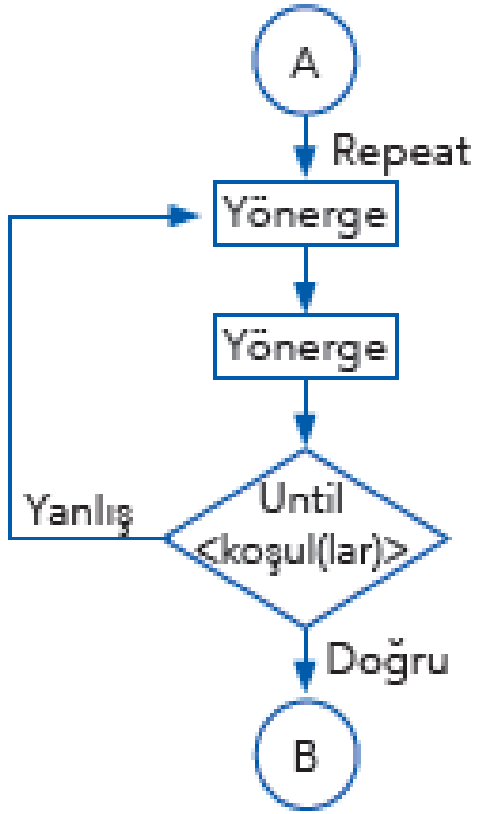
- Belirli bir iş bir çok kez tekrarlanacaksa, programda bu iş bir kez yazılır ve döngü deyimleriyle istenildiği kadar tekrar tekrar çalıştırılabilir.
- Bir döngüde, arka arkaya tekrarlanan deyimler döngü blokunu oluşturur. Bu deyimler birden çoksa { } bloku içine alınır. Bir döngü blokunda çok sayıda deyim olabileceği gibi, iç-içe döngüler de olabilir.
- Program akışının döngü blokunu bir kez icra etmesine döngünün bir adımı (bir tur) diyeceğiz.



Döngü Çeşitleri- Repeat-Until Döngüsü

- Repeat Until döngüsü belirteceğiniz bir şart doğru oluncaya kadar Repeat ile Until arasına yazacağınız bütün komutları tekrar tekrar başa giderek çalıştırır. Ta ki until ile belirttiğiniz şart doğru oluncaya kadar. Genel kullanımı:
- Repeat
- İşlem 1
- İşlem 2
- İşlem 3
- ...
- ...
- Until (sart);

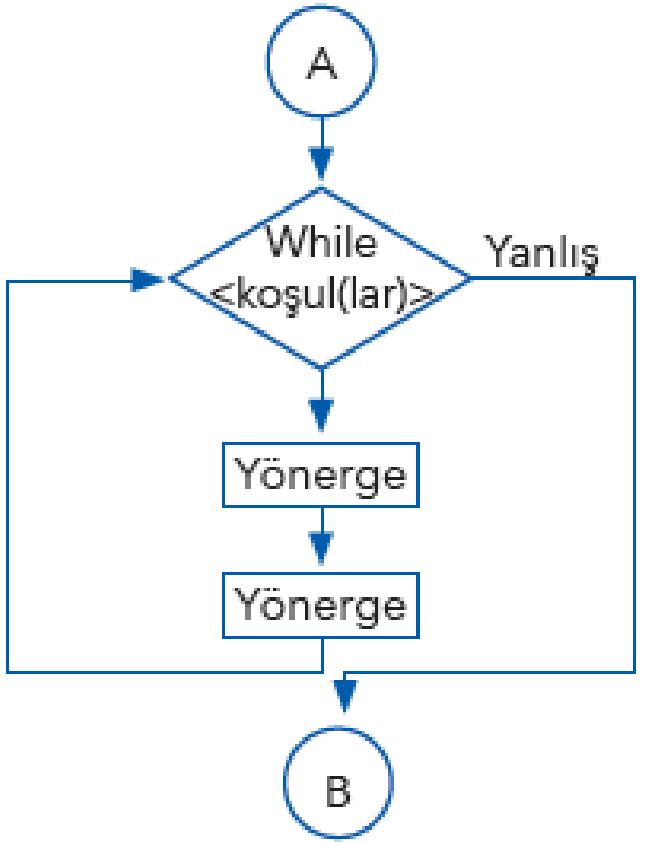




Döngü Çeşitleri- While Döngüsü

- Döngünün başlangıcında, döngü içerisinde yer alan yönergelerin işlenip işlenmeyeceğine ilişkin karar vermek için koşul kontrol edilir. Eğer koşul yanlış olursa döngü içerisindeki hiçbir yönerge işleme alınmaz. Eğer koşul doğru ise döngü içerisindeki tüm yönergeler çalıştırılır ve tekrar döngünün başına dönülür. Döngü, koşul durumu yanlış olana kadar devam eder.

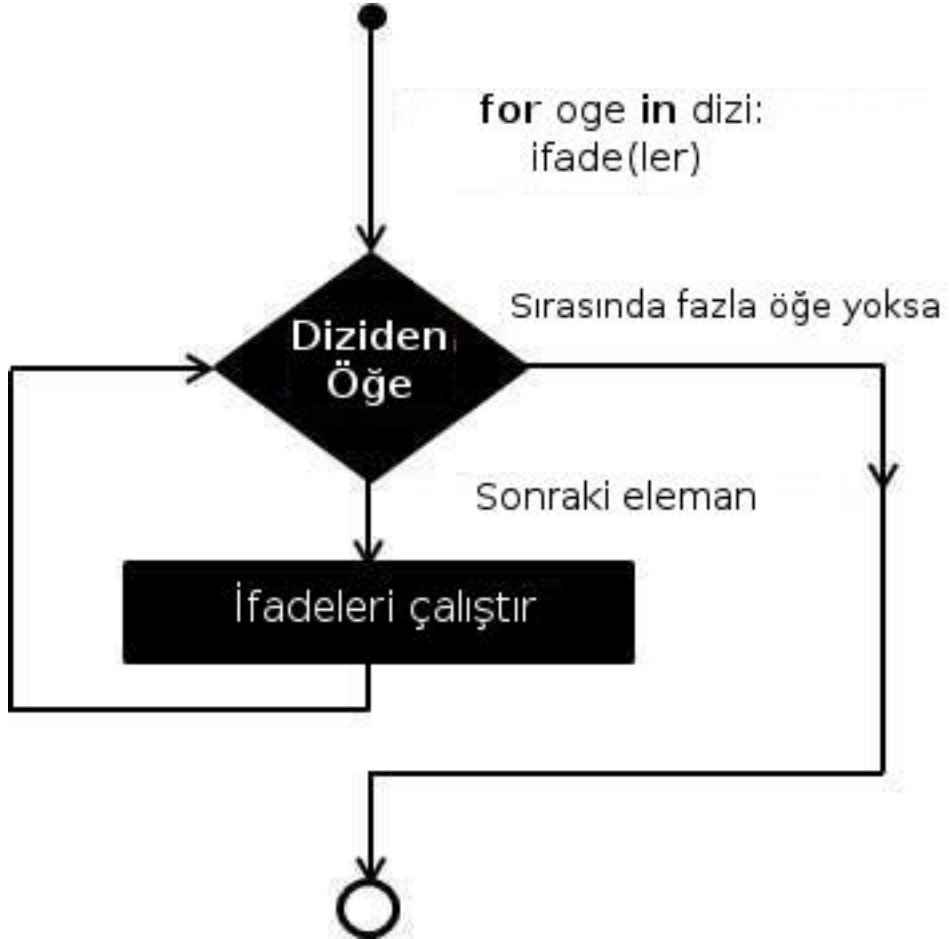
```
While<koşul(lar)>
Yönerge
Yönerge
……
WhileEnd
```



Döngü Çeşitleri- For Döngüsü

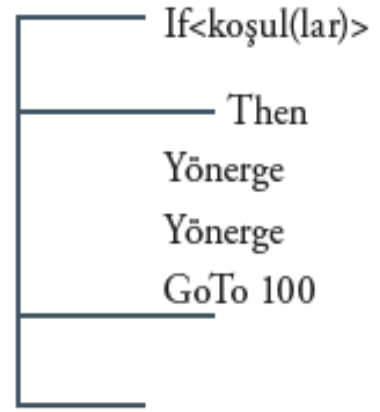
- *for deyimi* ve for deyimi kullanılarak oluşturulacak döngü yapısı, işlemlerin tekrar sayısının önceden belli olduğu durumlarda kullanılır.

```
for(ifade1;ifade2;ifade3)
        { Deyim1;
          Deyim2;
            ...
          Deyim_n;
        }
        Deyim_x;
```

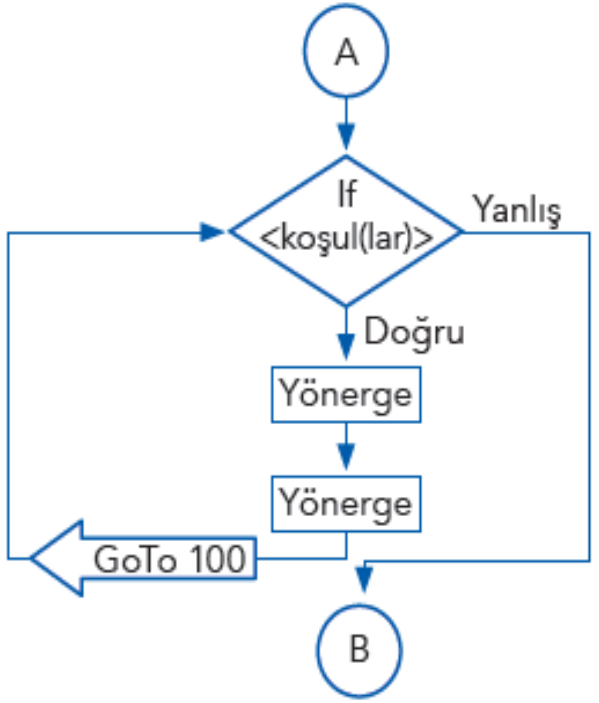


Döngü Bloklarına Döngü Dışından Gidilmesi

- Go To bir metod içerisinde bir işlemi yaptıktan sonra özellikle başka bir metoda gitmesi istenirs ya da bir switch case içerisinde tek değere göre birden fazla case gezdirmek gerekirse kullanılır.

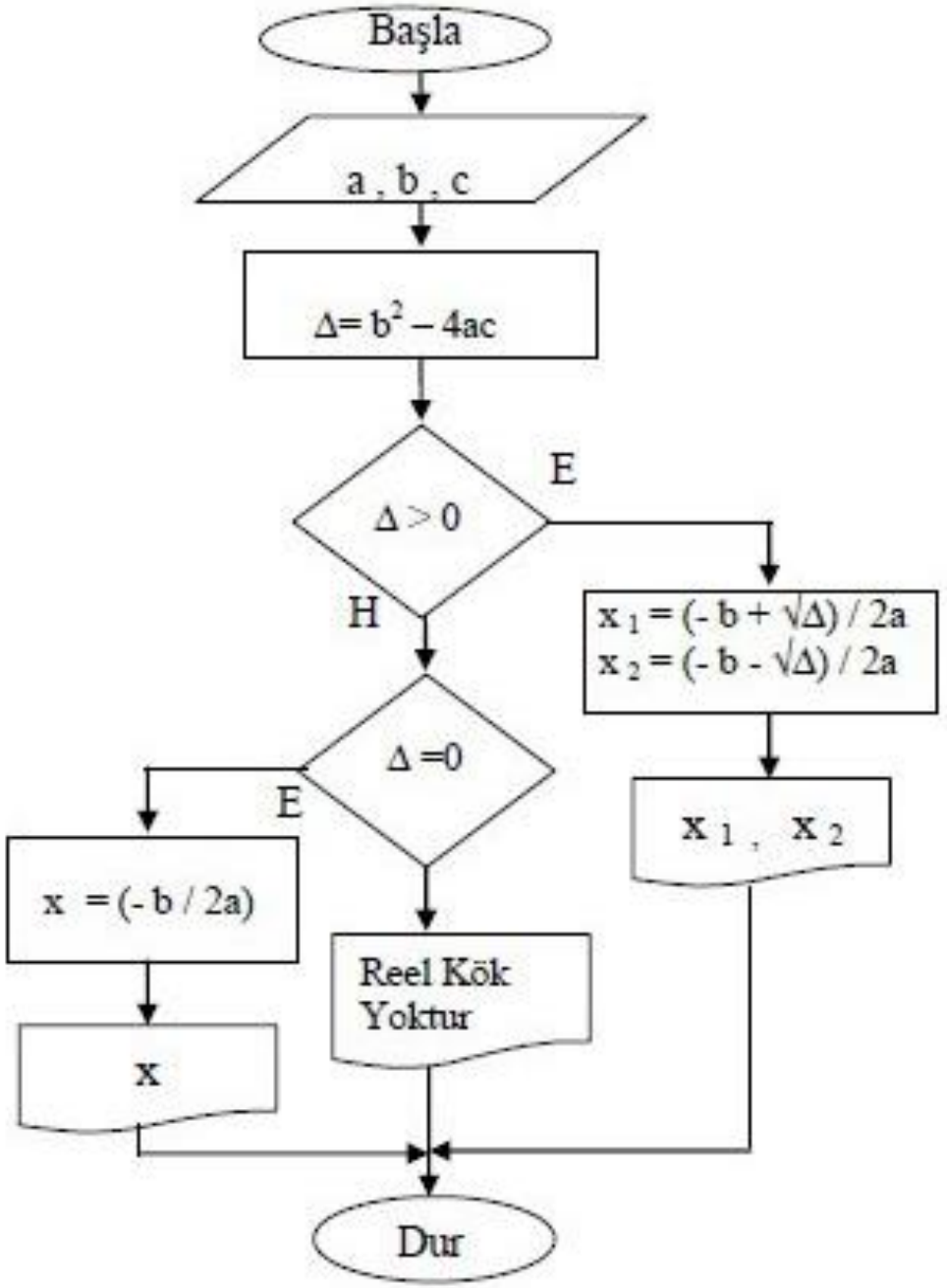



```
Etiketimiz:
int i = 0;
if (i == 0)
{
   MessageBox.Show("Cevabı Bildiniz");
   Close();
}
else
   goto Etiketimiz;
```



Akış Şeması Örneği

- 2. dereceden denklemin kökünü bulan akış şeması;



Algoritma ve İlişki Tanımlanması İçin Çeşitli Yöntemler

- Metinsel Tanımlama
- Akış Şeması
- N-S (Nassi-Schnederman) Şemaları
- W-O (Warnier-Orr) Şemaları
- --
- Bachman Notasyonu
- Crow’s Foot Spec Dili
- IDEF1XERD Spec Dili
- UML Yazılım Modelleme Dili
- SysML Sistem Modelleme

Sorularınız