Algoritma ve Programlaya Giriş II

Problem Çözme Sırası:

- Aşağıda ki adımları problem çözme sırasına göre doğru bir şekilde sıralayınız.
- *-*
- Algoritma ve Program Yazma
- Bir çözüm yolu geliştirme
- Test etme
- Problemi anlama

- 1- Problemi anlama
- 2- Bir çözüm yolu geliştirme
- 3- Algoritma ve program yazma
- 4- Test etme

Algoritmaların Sahip Olması Gereken Özellikler

- **1- Giriş/Çıkış Bilgisi:**
- Dışarıdan gelen verilere giriş bilgisi denir.
- Bu veriler algoritmada işlenir ve çıkış bilgisini oluşturur.
- Algoritmaların temel amacı giriş bilgisini işleyerek çıkış bilgisi oluşturmaktır.

- **2- Sonluluk:**
- Her türlü olasılık için algoritma sonlu adımda bitmelidir.
- Algoritma sonsuz döngüye girmemelidir.

- **3- Kesinlik:**
- Algoritmanın her adımı anlaşılır, basit ve kesin bir biçimde ifade edilmiş olmalıdır.
- Kesinlikle yorum gerektirmemeli ve belirsiz ifadelere sahip olmamalıdır.

- **4- Etkinlik:**
- Yazılan algoritmalar etkin ve dolayısıyla gereksiz tekrarlardan uzak oluşturulmalıdır.
- Ayrıca algoritmalar genel amaçlı yazılıp yapısal bir ana algoritma ve alt algoritmalardan oluşturulmalıdır.

- **5- Başarım ve Performans:**
- Birim İşlem Zamanı
- Veri Arama ve Getirme Zamanı
- Kıyaslama Zamanı
- Aktarma Zamanı

Sayının işaret durumuna göre işlem yapan algoritma

- 1- Başla
- 2- Sayı girilsin
- 3- Eğer sayı negatif ise o sayının karesi alınsın
- 4- Eğer sayı pozitif ise o sayının küpü alınsın
- 5- İşlem yapılan sayı yazdırılsın
- 6- Bitir

- Bir önceki sayfa da gördüğünüz algoritma kesinlik şartını sağlamıyor dolayısıyla bu algoritma kullanıma elverişli değildir çünkü hatalı sonuç üretebilir.
- Kesinlik şartının sağlanmamasının sebebi ise kullanıcı nötr bir sayı üzerinde işlem yapmak istediğinde algoritmanın yetersiz kalması.

Faktöriyel işlemini yapan algoritma

- 1- Başla
- 2- Faktöriyelini almak istediğiniz sayıyı giriniz
- 3- Girilen sayıyı birer birer azaltarak bir önceki sayı ile çarpınız
- 4- İşlem sonunda çıkan sonucu yazdırınız
- 5- Bitir

- Bir önceki algoritma kullanıma elverişli değildir çünkü sonsuz döngüye girecektir.
- Sonsuz döngüye girmesinin sebebi bir bitiş değeri konmaması. Azaltılan sayı 1’ e gelince işlemin sonlandırılsın komutu girilmeliydi.

Algoritmaların Gösterim Şekilleri

- 1- Düz yazı ile gösterimi
- 2- Sözde kod(pseudocode) ile gösterim
- 3- Akış şeması ile gösterim

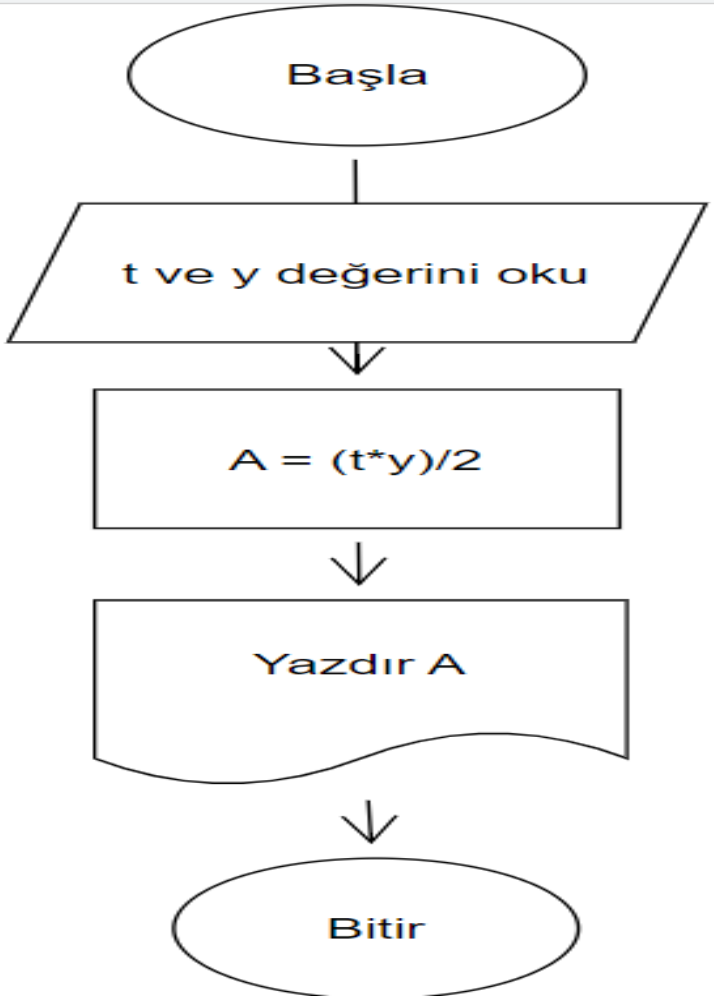
Üçgenin Alanını Hesaplayan Algoritma(Düz yazı ile gösterimi)

- 1- Başla
- 2- Taban değerini gir
- 3- Yükseklik değerini gir
- 4- Taban ile yüksekliği çarp ve sonucu ikiye böl
- 5- Çıkan sonucu yaz
- 6- Bitir

Üçgenin Alanını Hesaplayan Algoritma(Sözde kod ile gösterimi)

- Taban için t, yükseklik için y, alan için A seç
- 1- Başla
- 2- t değerini oku
- 3- y değerini oku
- 4- A = (t*y)/2
- 5- A değerini yaz
- 6- Bitir

Üçgenin Alanını Hesaplayan Algoritma(Akış şeması ile gösterimi)



İşlemler ve Operatörler

- 1- Matematiksel İşlemler
- 2- Karşılaştırma İşlemleri
- 3- Mantıksal (logic) işlemler

1- Matematiksel(Aritmetik) Operatörler

+ : Toplama
- : Çıkarma
* : Çarpma
/ : Bölme
^ : Üs Alma (n^)
% : Mod Alma

2- Karşılaştırma Operatörleri

< : Küçük
> : Büyük
<= : Küçük Eşit
>= : Büyük Eşit
== : Eşit
!= : Eşit Değil

3- Mantıksal(logic) Operatörleri

Ve : And, &

Veya : Or, ||

Boolean

- True ya da False olmak üzere iki çeşittir.
- Eğer koşul sağlanmış ise True yani 1 döndürülür.
- Ama koşul sağlanmamış ise False yani 0 döndürülür.

Algoritma da Kullanılan Temel Kavramlar

- 1- Tanımlayıcı
- 2- Değişken
- 3- Atama
- 4- Sayaç
- 5- Döngü

Tanımlayıcı

- → Programcı tarafından programdaki değişkenleri, sabitleri, alt programları vb. adlandırmak için kullanılan ifadelerdir.
- 1- İngilizce harfler kullanılır.
- 2- Simgelerden sadece (_) kullanılır
- 3- Tanımlayıcı isimler harfle veya alt çizgiyle başlayabilir.
- 4- Tanımlayıcı ismi, rakamla başlayamaz veya sadece rakamlardan oluşamaz.

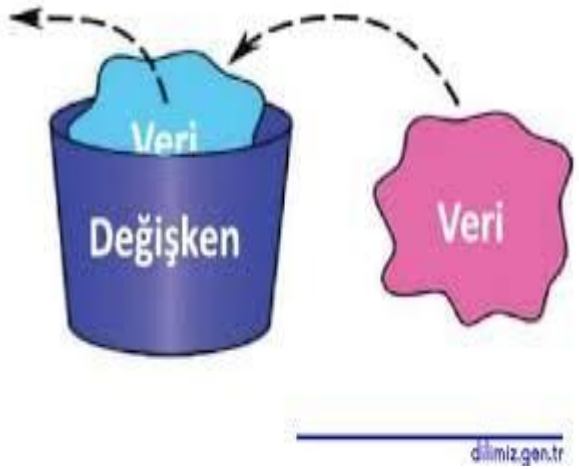
Aşağıdakilerden hangisi tanımlayıcı kurallarına uygun değildir?

- Doğukan
- _final
- ateş&barut
- 3silahşör

- **doğukan;** tanımlayıcı ifadesine uygun değil çünkü türkçe karakter içermektedir.
- **_final;** tanımlayıcı ifadesine uygundur
- **ateş&barut;** tanımlayıcı ifadesine uygun değildir çünkü geçersiz karakter kullanımı var
- **3silahşör;** tanımlayıcı ifadesine uygun değil çünkü sayı ile başlamış

Değişken

- Programın çalışması sırasında farklı değerler alabilen, değerleri değişebilen verilere denir.



Atama

- Değişkenlere değer aktarma işlemidir.
- Değişkenlere atanan bu değerler daha sonra tekrar kullanılabilirler.
- Sağdaki değer sonucu değişkene aktarılır. Bu durumda değişkenin bir önceki değeri varsa silinir.
- Atama yapmak için ise eşittir (=) operatörü kullanılır.

Değişken = Değer

i = 0

- 1- Başla
- 2- A = 5 atamasını yap
- 3- Yazdır A
- 4- A = «Algoritma» atamasını yap
- 5- Yazdır A
- 6- Bitir

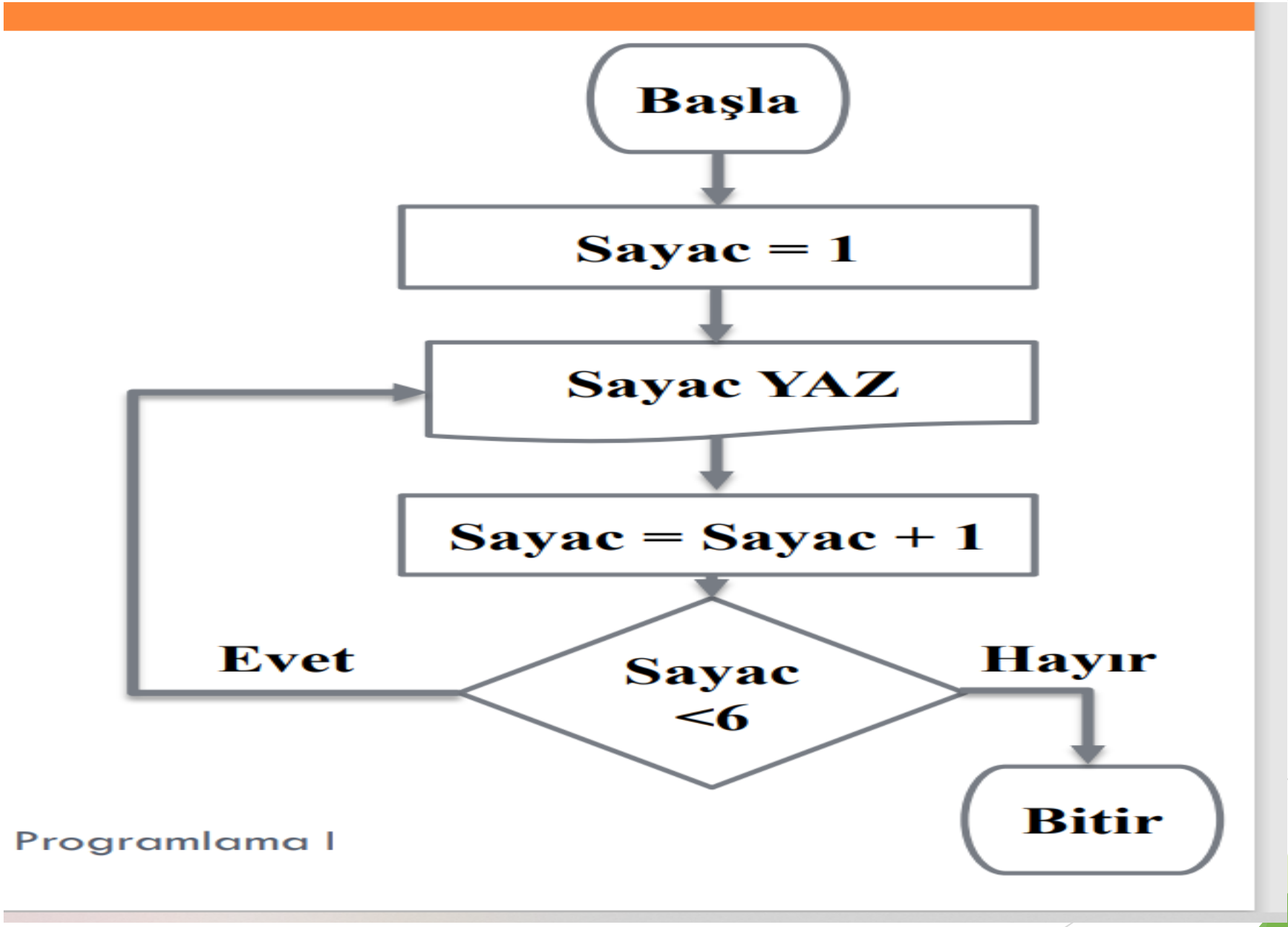
- Biraz önceki algoritma da ilk olarak ekrana 5 değerini sonrasında ise «algoritma» kelimesini bastırır çünkü değişkenler geçici olarak değer tutma özelliğine sahiptir. Aynı anda birden fazla değeri tutamaz.

Sayaç

- Bazı işlemlerin belirli sayıda yaptırılması ve üretilen değerlerin sayılması gerekebilir.
- Bu tür sayma işlemlerine algoritmada Sayac adı verilir.
- Sayaçlar da birer değişkendir.
- Ör: Sayac = Sayac + 1

1-5 arasındaki sayıların ekrana yazdırılması(söde kod ile gösterimi)

- 1- Başla
- 2- Sayac = 1
- 3- Sayac değerini yazdır
- 4- Sayac = Sayac +1 işlemini yap
- 5- Eğer Sayac < 6 ise 3. adıma git
- 6- Bitir


Başla
Sayac = 1
Sayac YAZ
Sayac = Sayac + 1
Sayac <6
Evet
Hayır
Bitir

Aşağıdaki sorular True mu False mu döndürür cevaplayalım

- x = 5 ataması yapılsın.

- 1) x != 7 koşulu sorgulansın
- 2) x>= 8 koşulu sorgulansın
- 3) x == 5 koşulu sorgulansın
- 4) x < -2 koşulu sorgulansın

- x = 5 ataması yapılsın.

- 1) x != 7 (True döndürür)
- 2) x>= 8 (False döndürür)
- 3) x < -2 (False döndürür)
- 4) x == 5 (True döndürür)

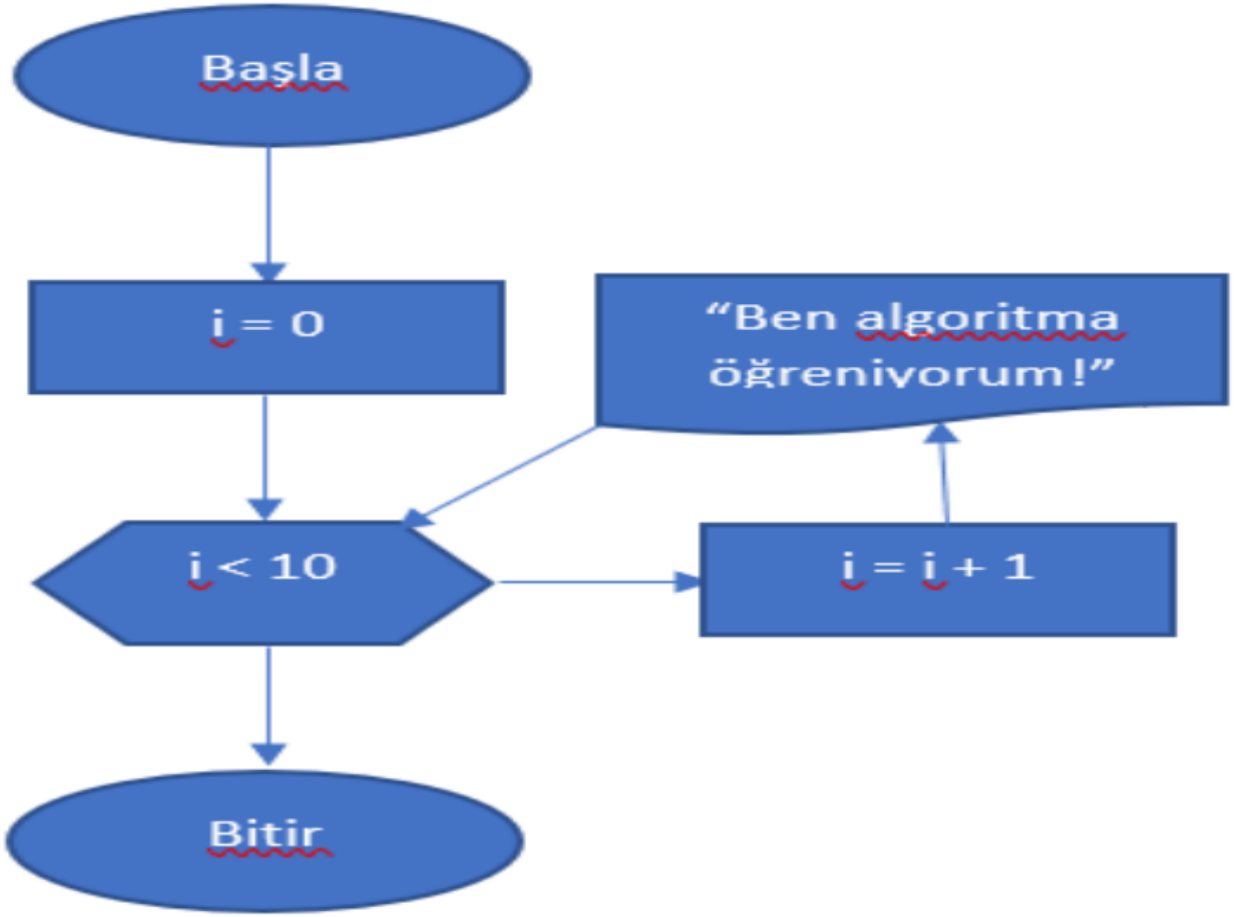
Döngü

- Tekrar eden işlemleri gerçekleştirmek için tasarlanmış yapılardır.
- Algoritmada yaygın olarak kullanılan 2 döngü yapısı vardır;
- 1- While döngüsü
- 2- For döngüsü
- Bonus: İç içe döngü

1- While Döngüsü

- While döngüsünde, döngü yapısının içinde bir koşul vardır. Bu koşul sağlandığı sürece döngüye bağlı olan işlemler tekrarlanır.

- Örneğin while döngüsü ile 9 kere ekrana «Ben algoritma öğreniyorum!» yazdıran programın algoritmasını oluşturalım.

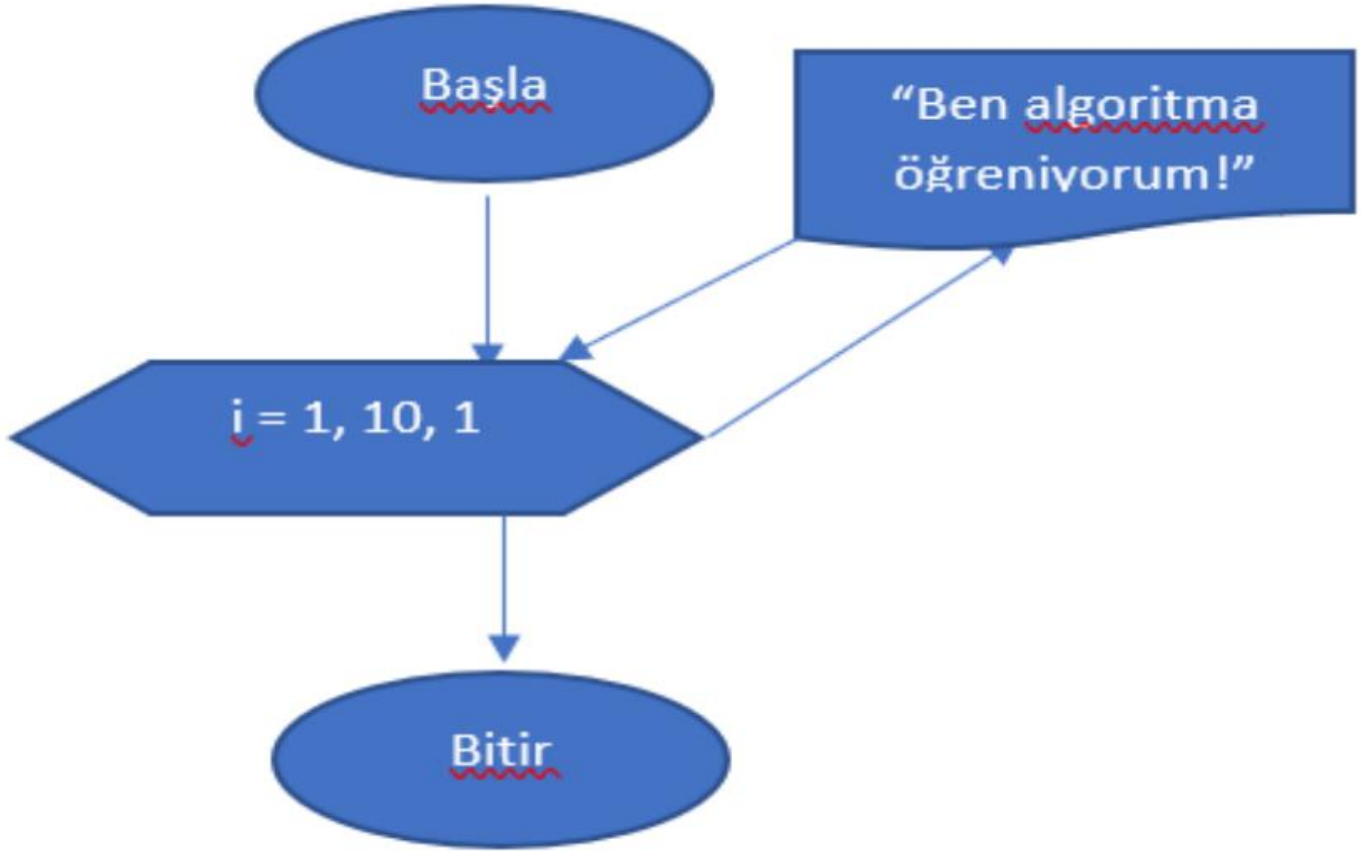


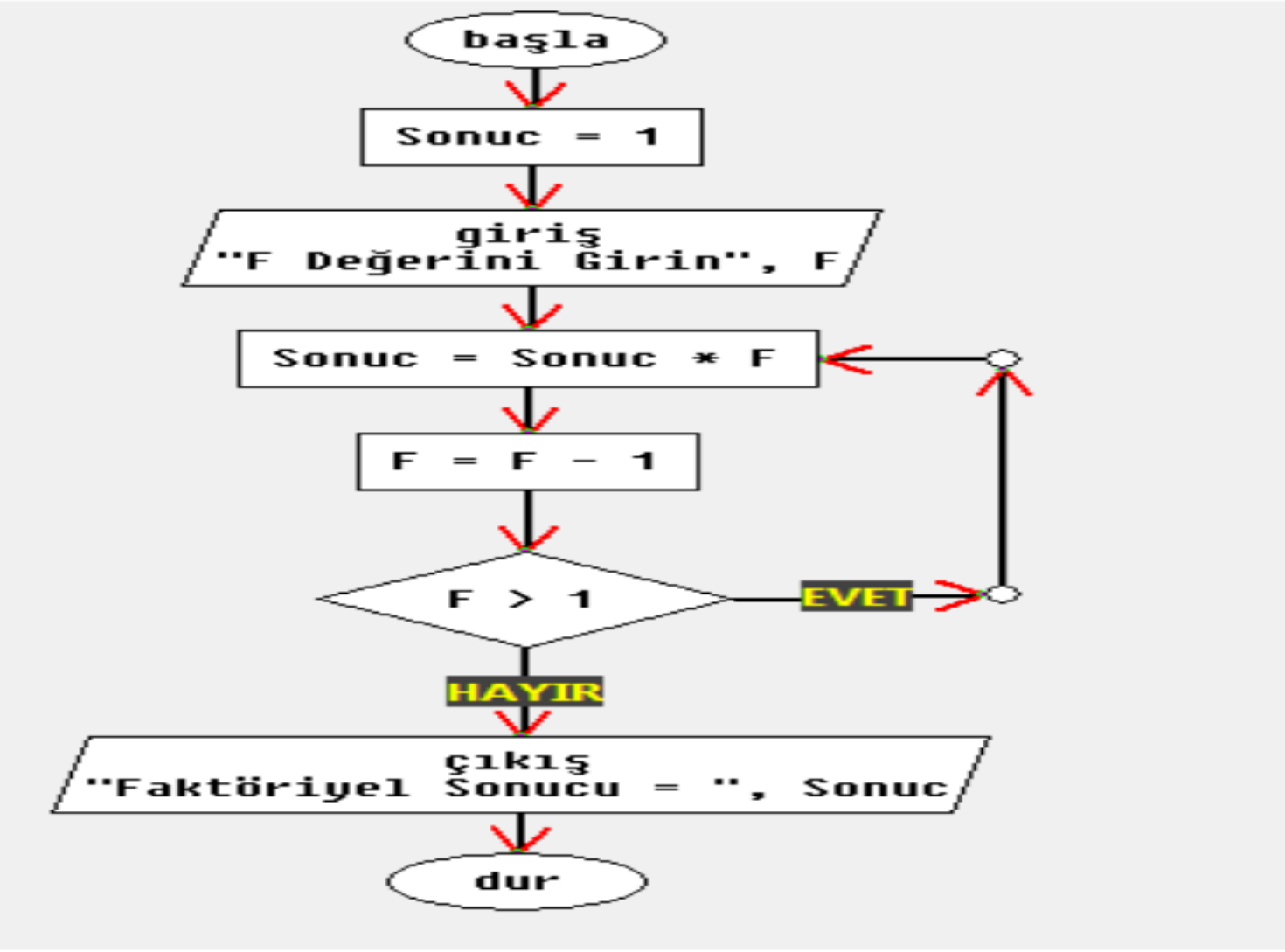
2- For Döngüsü:

- Değişken döngü yapısının içinde belirtilir ve bu döngü yapısının içindeki değişkene birbirinden virgülle ayrılmış 3 adet tamsayı değeri atanır.

- Bu değerlerin ilki döngüdeki değişkenin başlangıç değerini belirtir.
- İkincisi ise değişkenin ulaştırılması hedeflenen değeri belirtir.
- Sonuncu değer ise döngünün her tekrarında değişkene eklenecek değeri belirtir.

- Örneğin for döngüsü ile 9 kere ekrana «Ben algoritma öğreniyorum!» yazdıran programın algoritmasını oluşturalım:




başla
Sonuc = 1
giriş
"F Değerini Girin", F
Sonuc = Sonuc * F
F = F - 1
F > 1
EVET
HAYIR
çıkış
"Faktöriyel Sonucu = ", Sonuc
dur

İç İçe Döngüler

- Eğer bir döngüye bağlı başka bir döngü varsa bu döngülere iç içe döngü denir. İç içe döngülerde bir döngünün her tekrarında bağlı olan döngü baştan sona tekrarlanır.

Yıldızlarla kare bastırma