PROGRAMLAMA DİLLERİ

Bilgisayarları Kullanabilmek için onlarla iletişim kurmak gerekir. Bu iletişimi kurabilmek programlamanın amacıdır.

Program, bilgisayara bir dizi iş yaptıran komutlardan oluşur. Oluşturulan bir çözüm mantığının içerisindeki işlem aşamalarının çeşitli komut ve deyimlerle kodlanarak bilgisayarın anlayacağı bir dile çevrilmesi gerekir. Bu aşamada programlama dilleri denilen kavramlar ortaya çıkmaktadır.

ALGORİTMA

Bilgisayarlarda, bir problemin çözümünde izlenecek yol genel tanımıyla algoritma olarak adlandırılır.

Algoritmalar, bir problemin çözümündeki işlemlerin, kararların ve bunların icra edildiği sıranın oluşturduğu akış olarak düşünülebilir. Algoritma kurma, programlama aşamasının en önemli kısmıdır. Burada üretilen mantıksal akışlar, bir anlamda programlama olarak adlandırılan ve bilgisayarın anlayabileceği dilde kodlama sisteminin temelini oluşturmaktadır.

Programlamaya başlamadan önce, problemin çözümüne ilişkin mantıksal akışları içeren algoritmaların oluşturulması, programlama aşamasında son derece kolaylık sağlayacaktır. Geriye sadece ilgili programlama dilinde işlemleri kısaltıcı fonksiyonların kullanılması ve kodlama işlemlerinin yapılması kalacaktır.

Bilgisayar programlaması sırasında izlenebilecek bir çok yol ve yöntem vardır. Programcının probleme ilişkin çözümü ortaya çıkarabilmesi için problem çözümü ile ilgili bilgileri bilmesi gerekir. Bilgisayar programlamasında genel olarak belirli kalıp ve kurallara uyulur. Bir bilgisayar yazılımının oluşturulması sırasında aşağıda sıralanan adımlara uyulur.

* Problemin tanımı
* Çözüm yönteminin belirlenmesi
* Programın kodlanması
* Programın çalışır duruma getirilmesi
* Belgeleme ve güncelleştirme

Problemin Tanımı: Problemin normal yazı diliyle tanımlanması işlemlerini kapsamaktadır. Problem çözümüne ilişkin iyi bir program yapabilmek için, problemin iyi bir şekilde tanımlanması gerekir.

Çözüm Yönteminin Belirlenmesi: Bu adımda çözümün genel yaklaşımı, temel giriş/çıkışlar belirlenir ve problem çözümü adım adım program akış diyagramlarıyla gösterilir.

Programın Kodlanması: Program ayrıntılı olarak tanımlanıp çözüm yolları açıkça belirtildikten sonra program kodlama çalışmalarına başlanabilir. Programın baştan sona yapısal bir düzende hazırlanması ve uygun bir programlama dili seçilmesi seçim işleminin ilk aşamasını oluşturur.

Programın Çalışır Hale Getirilmesi: Programın Kodlanması sırasında yapılan imla hataları, kodlama ve mantık hatalarının giderilmesi işlemlerini kapsar. İyi bir bilgisayar programının doğruluğundan emin olmak için defalarca test edilmiş olması gerekmektedir.

Algoritma Kurma

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 sıralanmalıdır. 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:

1-Evden dışarıya çık ve otobüs durağına yürü
2-Durakta gideceğin yöndeki otobüsü bekle
3-Otobüs geldiğinde otobüse bin biletini bilet kumbarasına at
4-İneceğin yere yakınlaştığında arkaya yürü
5-İneceğini belirten ikaz lambasına bas
6-Otobüs durunca in, işyerine doğru yürü, iş yeri giriş kapısından içeri gir
7-Mesai arkadaşlarınla selamlaş
8-İş giysini giy
9-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 kaçırma olasılığı düşünülmeden oluşturulmuştur. Kişi durağa geldiğinde bineceği otobüsü kaçırmış ise algoritmamız aşağıdaki şekilde değiştirilebilir.

Çözüm 2:

1-Evden dışarıya çık
2-Otobüs durağına yürü
3-Otobüsün saati geçmiş mi?
4-Durakta gideceğin yöndeki bir sonraki otobüsü bekle
5-Bir sonraki otobüs gelene kadar 4. adımı uygula
6-Otobüsün geldiğinde otobüse bin
7-Biletini bilet kumbarasına at İneceğin yere yakınlaştığında arkaya yürü
8-İneceğini belirten ikaz lambasına bas
9-Otobüs durunca in
10-İşyerine doğru yürü
11-İş yeri giriş kapısından içeriye gir
12-Mesai arkadaşlarınla selamlaş
13-İş giysini giy işini yapmaya basla.

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.

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

Algoritma:

A1 :Birinci sayıyı gir
A2 :İkinci sayıyı gir
A3 :İki sayının toplamını yap
A4 :Toplamın değerini yaz
A5 :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 aşağıdaki yöntem izlenmelidir:

DEĞİŞKEN KAVRAMI

Bir problemin çözümünde tanımlanan bir bilgi alanı, farklı adımlarda farklı değerler alabiliyorsa bu bilgi alanına değişken adı verilir. Tanımlanacak bir değişken için aşağıdaki kurallara dikkat etmek gerekmektedir.

1- *Bir değişken adı A ile Z arasındaki alfabetik harfler ile başlamalıdır. Değişken adı, bir kelime yada arada boşluk olmama koşulu ile bir cümle olabilir.*

A, TOPLAM, SAYI, SONUC, ADSOYAD gibi tanımlanabilir.

2- *Bir değişken adının ilk karakteri sayısal olamaz, yani 0 ile 9 arasında bir rakam ile başlayamaz. Ancak, ilk karakterden sonra istenilen bir sayı kullanılabilir.*

Al, TOPLAM1, K1A37, B1236, ... şeklinde kullanılabilir.

3- *Değişken adı algoritmanın kodlanacağı programlama diline ait bir komut yada deyim olamaz.*

PRINT, END, NO, READ, ... şeklinde kullanılamaz,

4- *Algoritmada değişken adı verilirken Türkçe karakterler kullanmamaya dikkat edilmelidir.*

SONUÇ, DEĞER, KOŞUL, ... gibi.

AKTARMA VE ATAMA İŞLEMLERİ

Bir aritmetik ifadenin yada bir değerin herhangi bir değişkene tanımlanmasına aktarma yada atama işlemleri adı verilir. Bu atama ve aktarma işlemleri sayısal ifadeler ve aritmetik ifadeler için;

<değişken adı>=<aritmetik ifade yada değer> şeklindedir.

Eğer atanacak ifade sayısal ifade değilse bu durumda ifade “ ile birlikte atanmak zorundadır.

<değişken adı>= “sayısal olmayan ifade” şeklindedir,

Örnek olarak, TOPLAM=A+B, A=3, ISIM=”AHMET” verilebilir.

ARTIRIM İŞLEMLERİ

Bilgisayar mantığında matematik mantıktan farklı olarak bazı işlemler yapılabilmektedir. Herhangi bir değişkene kendisi ile birlikte bir değeri atamak mümkündür. Örnek olarak,

TOPLAM=TOPLAM ± A tanımlaması yapılabilir.

Burada, eşitliğin solunda tanımlanan yeni TOPLAM ifadesi, önceki TOPLAM ifadesine A ilave edilerek elde edilmiştir. Benzer şekilde;

X=X+ 1, SAYAC=SAYAC-1, SAYI=SAYI-5-A-1
gibi tanımlamalar yapılabilir.

ARİTMETİK İŞLEMLER

Algoritma ve Akış Şemalarında kullanılacak olan işlem operatörleri aşağıdaki gibi tanımlanmaktadır.

(+) Toplama
(-) Çıkarma
(*) Çarpma
(/) Bölme
(=) Aktarma ve Eşitlik
(^) Üs Alma

(<>) Eşit Değil (Farklı)
(<) Küçüktür
(>) Büyüktür
(<=) Küçük yada Eşit
(>=) Büyük yada Eşit

Matematiksel işlemlerdeki operatörlerin kullanım öncelikleri, algoritma ve akış şemalarında da aynen geçerlidir.

Bu kullanım öncelikleri aşağıdaki gibi tanımlanmaktadır.

1. Öncelikli, varsa parantez içerisi ( )
2. Öncelikli, Üs alma işlemi
3. Öncelikli, Çarpma ve Bölme işlemi *, /
4. Öncelikli, Toplama ve Çıkarma işlemi, +, -

Örnek 2 deki 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 işlemler 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 **T,** Birinci Sayı için **X,** İkinci Sayı için **Y** değerleri kullanılırsa;

Algoritma:

A1 : X değerini gir
A2 : Y değerini gir
A3 : T = X+Y
A4 : T 'yi yaz
A5 : 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:

A1 : X değerini gir
A2 : Y değerini gir
A3 : Z =X+Y
A4 : Ort=Z/2
A5 : Ort değerini yaz
A6 : 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:

A1 :Top =0, Sayaç =0
A2 :X değerini gir
A3 :Top=Top+X
A4 :Sayaç = Sayaç +1
A5 :Eğer Sayaç < 5 ise A2 'ye git
A6 :Ort=Top/5
A7 :Top ve Ort değerlerini yaz
A8 :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:

A1 : a değerini gir
A2 : a<0 ise 1. adımı tekrarla
A3 : b değerini gir
A4 : b<0 ise 3. adımı tekrarla
A5 : Alan =a*b
A6 : Alan değerini yaz
A7 : Bitir

Örnek 6: Verilen bir sayının faktöriyelini hesaplayan programın algoritmasının oluşturulması

Sayının faktöriyeli: **Fktrl**, Faktöriyel degiskeni: **X**,
Faktöriyeli hesaplanacak sayi : **Y**

Algoritma:

A1 : Fktrl=1, X=0
A2 : Y ‘yi gir
A3 : Y<1 ise 2. adimi tekrarla
A4 : X=X+1
A5 : Fktrl=Fktrl*X
A6 : X<Y ise 4. adıma geri dön
A7 : Fktrl değerini yaz
A8 : 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

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 Y ’ye eşit olması durumunda işlemler tamamlanarak hesaplanan değerin yazdırılması işleminden sonra programın çalışması sona ermektedir.

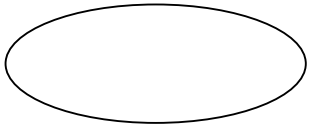
Akış Diyagramları

Geliştirilecek olan yazılımın genel yapısının şematik gösterimine ***akış diyagramı*** adı verilir. Akış Diyagramları, yazılımı oluşturacak program parçalarını ve bu parçaların birbirleri ile olan ilişkilerini belirler.

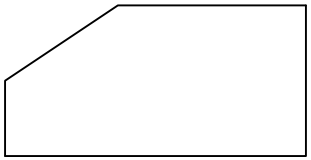
Bir bilgisayar programının oluşturulmasında akış diyagramlarının hazırlanması, algoritma oluşturma aşamasından sonra gelmektedir. Bilgisayar programının oluşturulması sırasında algoritma aşaması atlanarak, doğrudan akış diyagramlarının hazırlanmasına başlanabilir. Programlama tekniğinde önemli ölçüde yol almış kişiler bu aşamayı da atlayarak direkt olarak programın yazımına geçebilirler.

Akış diyagramlarının algoritmadan farkı, adımların simgeler şeklinde kutular içinde yazılmış olması ve adımlar arasındaki ilişkilerin (iş akışı) oklar ile gösterilmesidir.

Akış diyagramlarında kullanılan semboller, anlamları ve kullanış amaçları aşağıda verilmiştir.

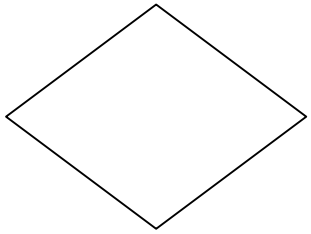


Algoritmanın başlangıç ve bitişinde kullanılır. başlangıç simgesinden çıkış oku vardır. Bitiş simgesinde giriş oku vardır.



Programa veri girişi ve programdan elde edilen sonuçların çıkış işlemlerini gösterir.

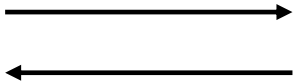
Aritmetik işlemler ve değişik atama işlemlerinin temsil edilmesi için kullanılır.



Aritmatiksel veya mantıksal olarak karar verme durumlarında kullanılır.

Yapılacak iş birden çok yapılacaksa (Döngü mantığı) bu sembol kullanılır

Herhangi bir değerin ekrana veya yazıcıya çıktısının alınacağı zaman kullanılır.



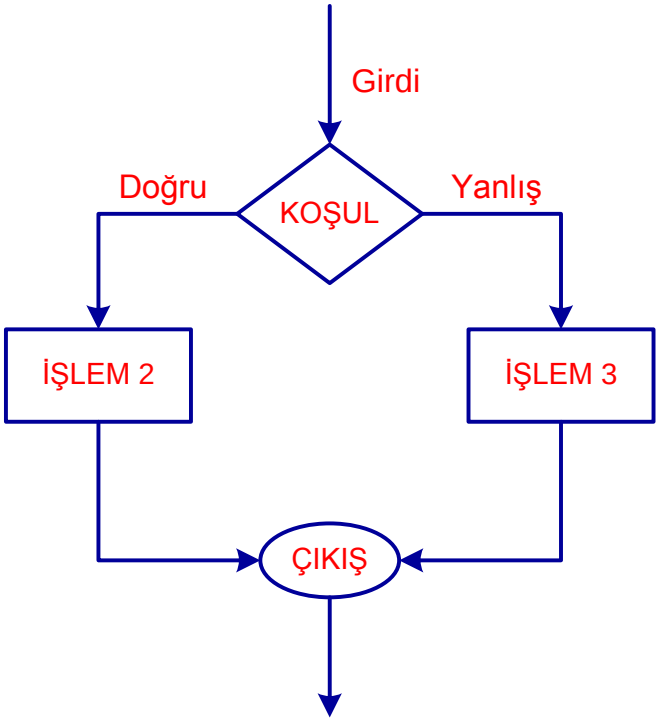
Diyagramın akis yönünü gösterir.

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***. Sıralı yapı, hazırlanacak programdaki her işlemin mantık sırasına göre nerede yer alması gerektiğini vurgular. Bu yapı sona erinceye kadar ikinci bir işlem başlayamaz.

Sıralı Yapı

Mantıksal yapılardan ikincisi ***Karar Verme*** yapısıdır. Programlama sırasında IF...Then... Else yapısı ile tanıyacağımız bu mantıksal yapılar, birden fazla sıralı yapı seçeneğini kapsayan modüllerde, hangi koşullarda hangi sıralı yapının seçileceğini belirler.



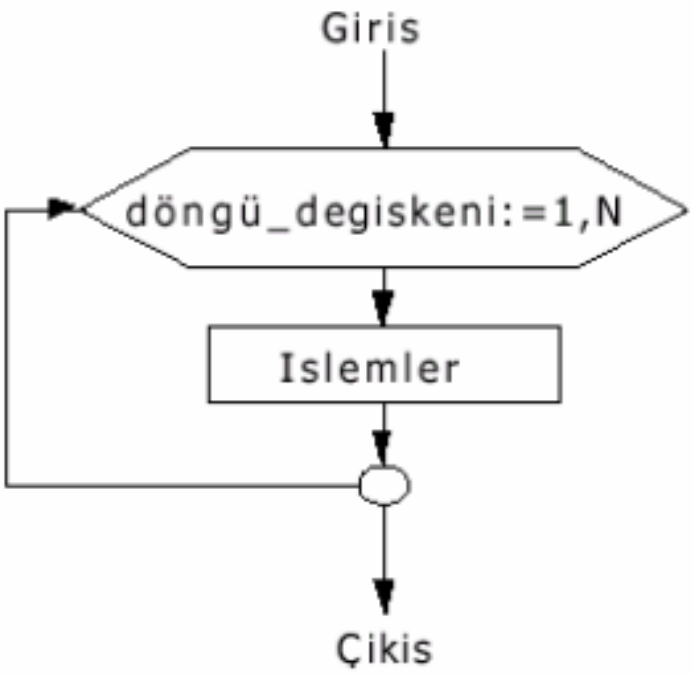
Karar Verme Yapısı

Üçüncü mantıksal yapı çeşidini **tekrarlı yapılar** oluşturmaktadır.

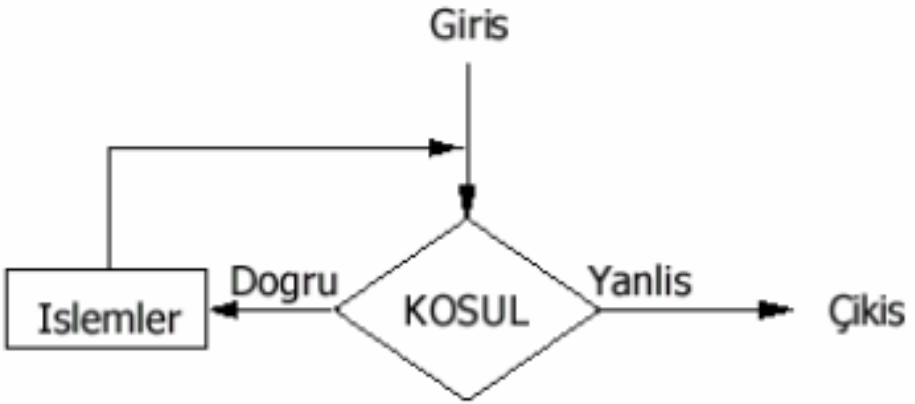
Bu yapılar çeşitli programlama dillerinde *For, While* ve *Repeat…Until* gibi komutlarla sağlanır. Şartlara göre değişik işlem gruplarının yapılmasını sağlar.

Bu yapı yukarıda sözü edilen iki yapının çeşitli birleşimlerinin tekrarlanmasından oluşmuştur.

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



a



b

Tekrarlı Yapılar

Sayılar ile ilgili algoritmalar ve akış diyagramları

Bu bölümde, sayıların değişik şekillerde kullanılmasına ilişkin algoritma ve akış şeması örnekleri yapılacaktır. İşlemlerde bir sayının tam kısmını alırken TAM ifadesi, mutlak değerini alırken de MUTLAK ifadesi kullanılacaktır. Örneğin A sayısının tam kısmı olarak TAM(A) ve mutlak değeri için de MUTLAK(A) ifadesi kullanılacaktır.

Örnek: Girilen iki sayının toplamını bulan programın algoritma ve akış şemasının oluşturulması.

Örnek: Girilen iki sayının toplamını bulan programın algoritma ve akış şemasının oluşturulması.

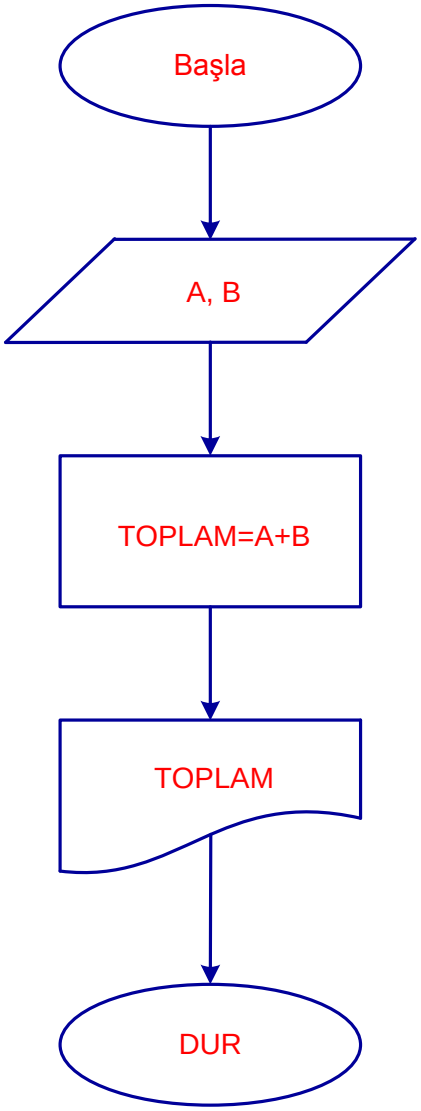
A1. Başla,

A2. A ve B sayılarını gir/oku,

A3. TOPLAM=A+B,

A4. TOPLAM'ı yaz,

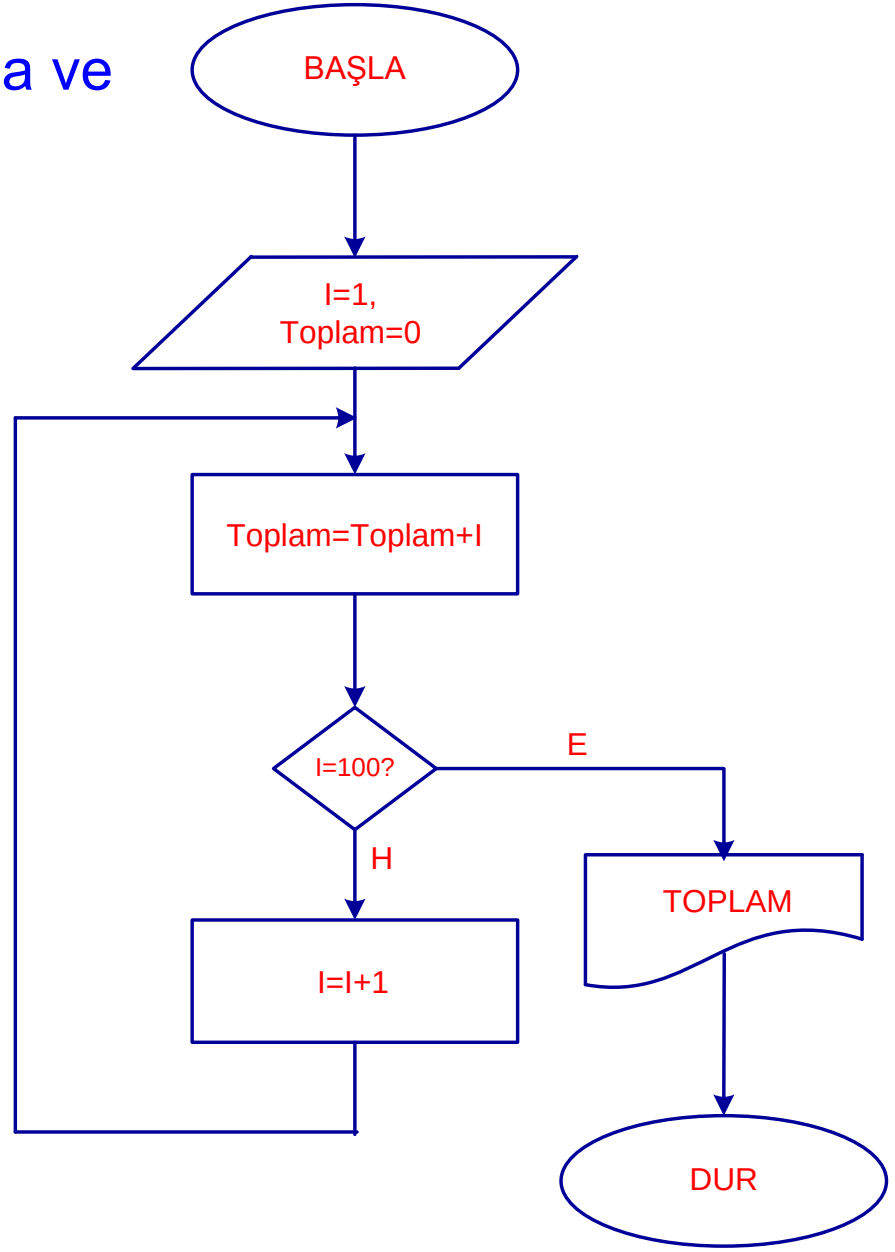
A5. Dur.



Örnek: 1 den 100 ’e kadar olan tamsayıların toplamını bulan algoritma ve akış şemasının oluşturulması.

Örnek: 1 den 100 ’e kadar olan tamsayıların toplamını bulan algoritma ve akış şemasının oluşturulması.

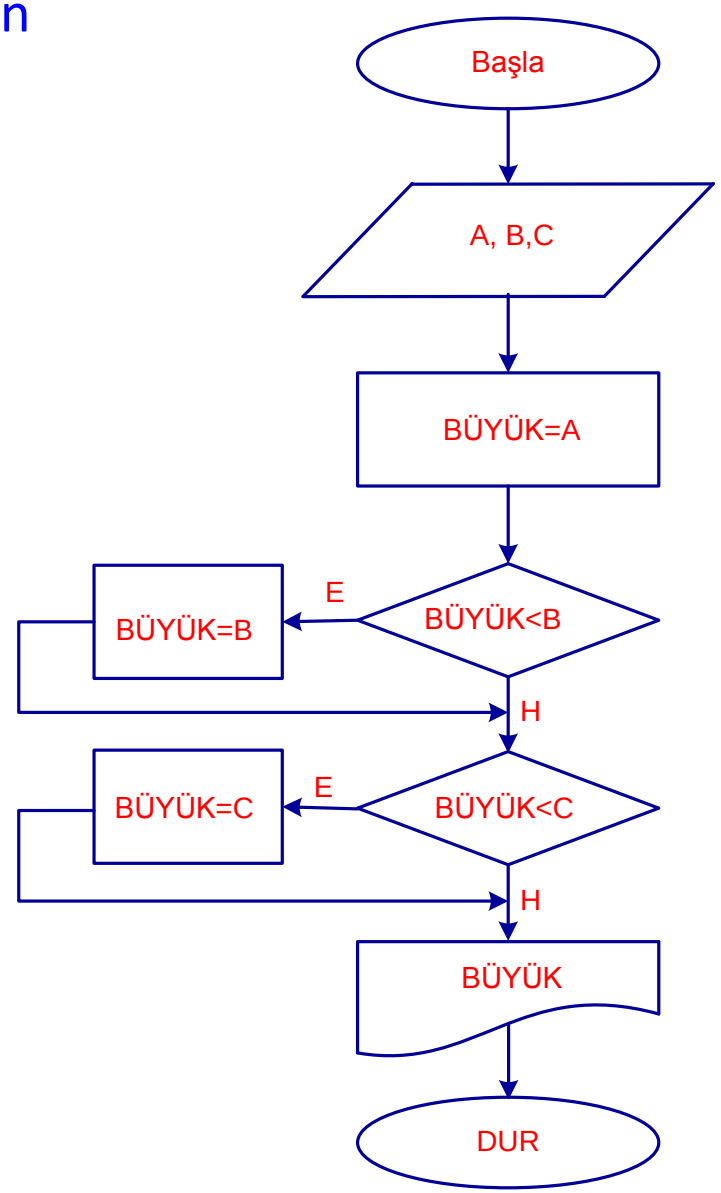
Al. Başla,
A2. i=1, TOPLAM=0,
A3. TOPLAM=TOPLAM+i,
A4. Eğer i= 100 ise A6. adıma git,
A5. i=i+1 al ve A3. adıma geri dön,
A6. TOPLAM değerini yaz,
A7. Dur.



Örnek : Girilen 3 tamsayıdan en büyüğünü bulan algoritma ve akış şemasının oluşturulması.

Örnek : Girilen 3 tamsayıdan en büyüğünü bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A, B ve C sayılarını gir,
A3. BUYUK=A,
A4. Eğer BUYUK<B ise BUYUK=B,
A5. Eğer BUYUK<C ise BUYUK=C,
A6. BUYUK değerini yaz,
A7. Dur.



ÖDEV:

1) ax+b=0 şeklinde verilen 1.derece denklemin çözümünü veren programa ait algoritma ve akis diyagramını hazırlayınız.

2) $ax^2+bx+c=0$ şeklinde verilen 2. derece denklemin köklerini bulan programın algoritma ve akış diyagramını hazırlayınız.

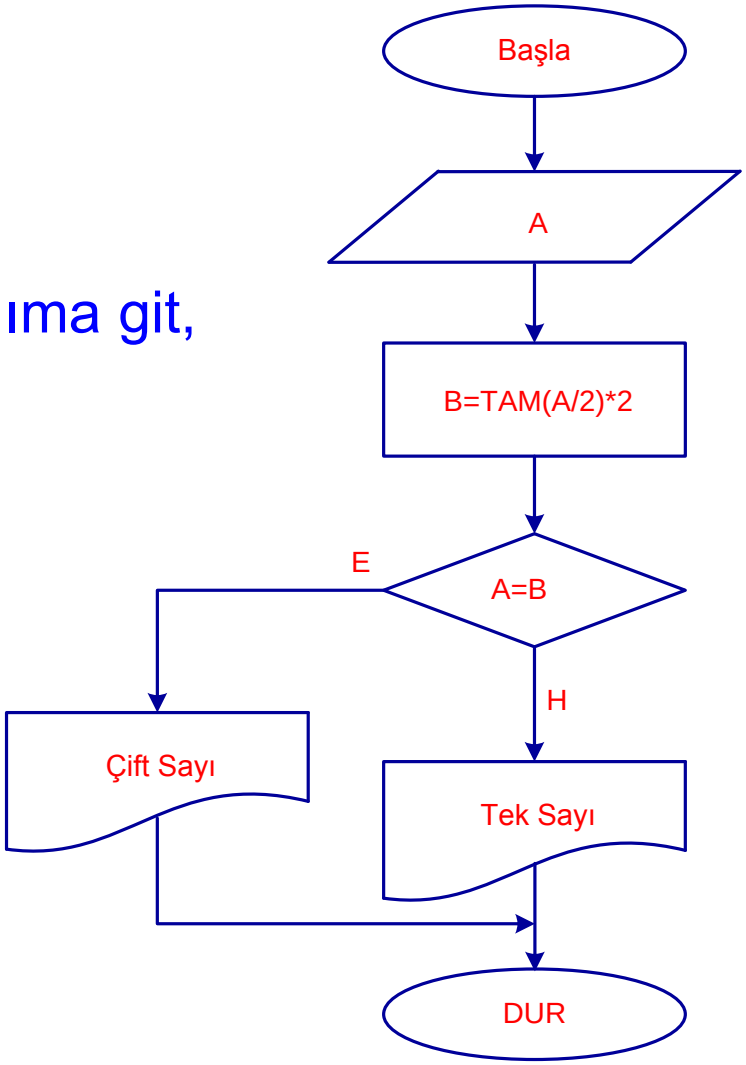
3) 10 tane N sayısının faktöriyelini hesaplayan programın algoritma ve akış diyagramını hazırlayınız.

4) Elimizde bulunan A, B, ve C gibi 3 adet sayıdan en büyüğünü ve en küçüğünü bulan programın algoritma ve akış diyagramını hazırlayınız.

Örnek: Girilen bir tamsayının tek yada çift olduğunu belirleyen algoritma ve akış şemasının oluşturulması.

Örnek: Girilen bir tamsayının tek yada çift olduğunu belirleyen algoritma ve akış şemasının oluşturulması.

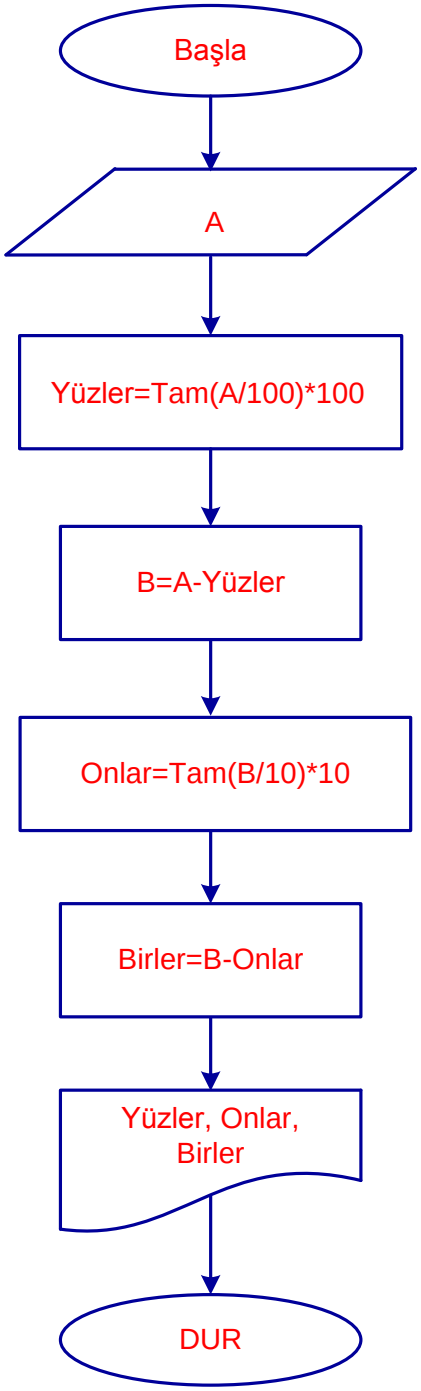
A1. Başla,
A2. A sayısını gir,
A3. B=TAM(A/2)*2,
A4. Eğer A=B ise A6. adıma git,
A5. ‘girilen sayı tek sayıdır” yaz ve A7. adıma git,
A6. ‘girilen sayı çift sayıdır” yaz,
A7. Dur.



Örnek : Üç basamaklı bir tamsayının birler, onlar ve yüzler basamağını bulan algoritma ve akış şemasının oluşturulması.

Örnek : Üç basamaklı bir tamsayının birler, onlar ve yüzler basamağını bulan algoritma ve akış şemasının oluşturulması.

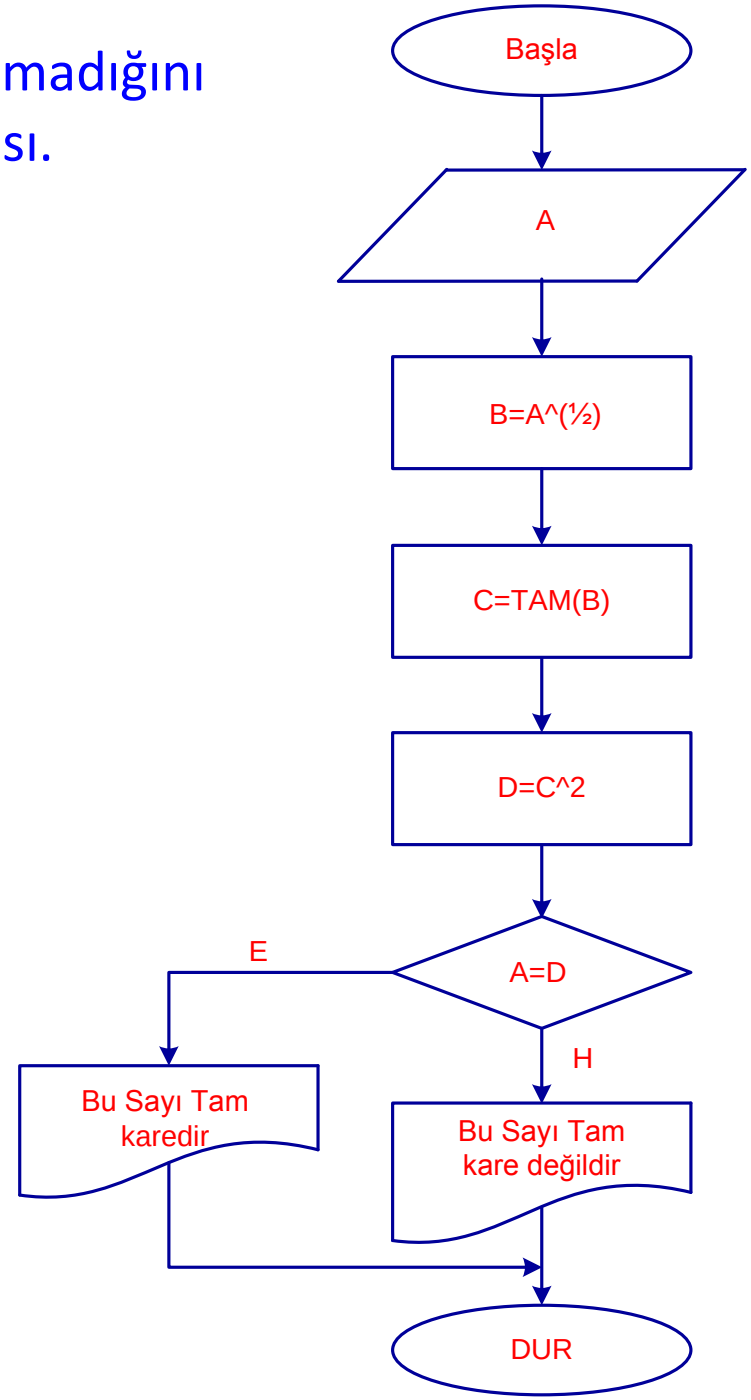
A1. Başla,
A2. A sayısını gir {Üç basamaklı bir sayı},
A3. YÜZLER=TAM(A/100)*100 ,
A4. B=A-YÜZLER ,
A5. ONLAR=TAM(B/10)*10 ,
A6. BIRLER=B-ONLAR ,
A7. YUZLER, ONLAR ve BİRLER değerlerini yaz,
A8. Dur.



Örnek : Girilen bir tamsayının tam kare olup olmadığını bulan algoritma ve akış şemasının oluşturulması.

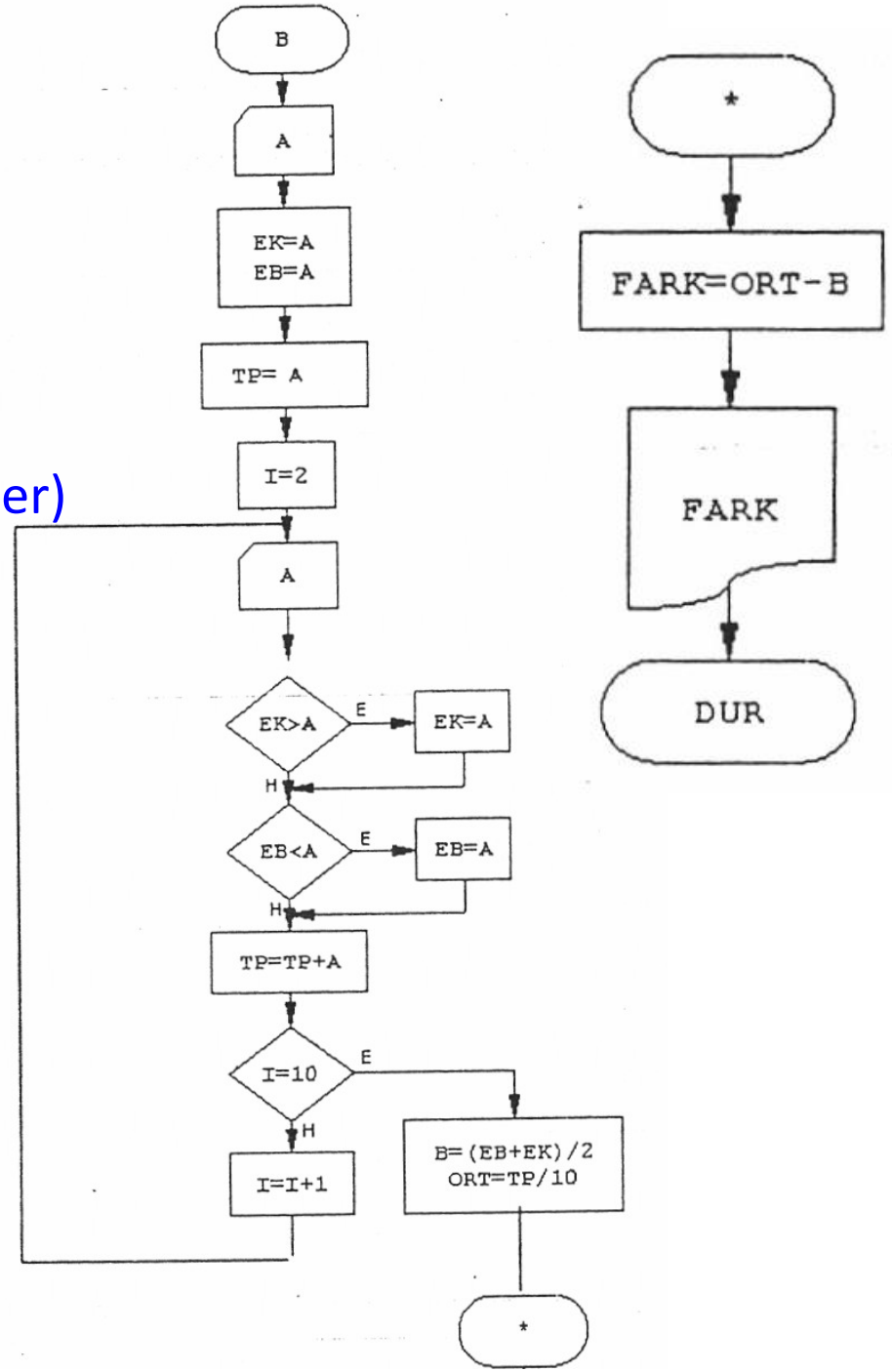
Örnek : Girilen bir tamsayının tam kare olup olmadığını bulan algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. B=A^(1/2) ,
A4. C=TAM(B) ,
A5. D=C^2 ,
A6. Eğer A=D ise A9. adıma git,
A7. “tam kare değil” yaz,
A8. A10. adıma git,
A9. “tam kare’’ yaz.
A10. Dur.



Örnek : Arka arkaya girilen rasgele 10 tamsayının ortalaması ile bu sayılardan en büyük ve en küçük olanının ortalamasını bularak elde edilen bu iki ortalamanın farkını alan algoritma ve akış şeması.

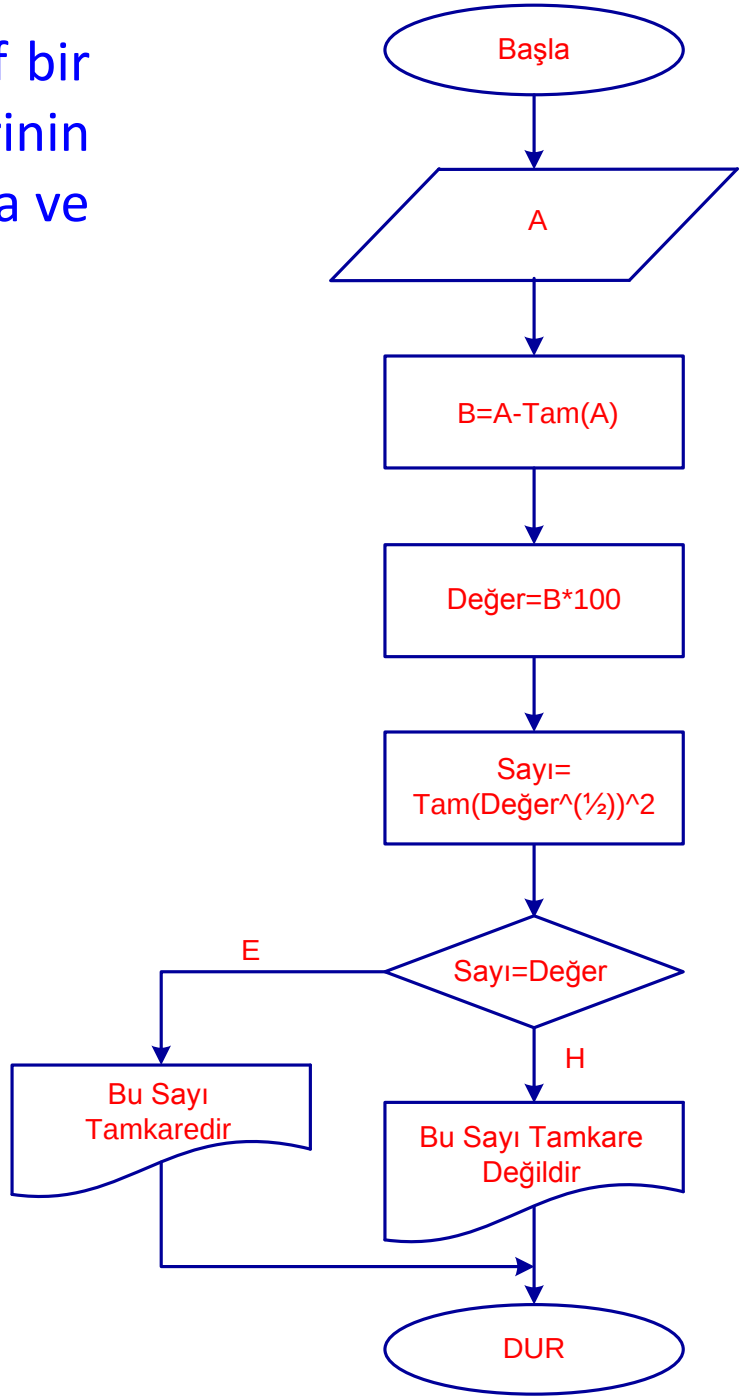
A1. Başla,
A2. A ’yı gir, (İlk sayının girişi)
A3. EK=A ,
A4. EB=A ,
A5. TP=A ,
A6. I=2 ,
A7. A ’yı gir, (İkinci sayıdan itibaren girişler)
A8. Eğer EK>A ise EK=A ,
A9. Eğer EB<A ise EB=A ,
A10. TP=TP+A ,
A11. Eğer I=10 ise A13. adıma git,
A12. I=I+1 al ve A7. adıma geri dön,
A13. B=(EB+EK)/2 ,
A14. ORT=TP/10 ,
A15. FARK=ORT-B ,
A16. FARK ’ı yaz,
A17. Dur.



Örnek : Ondalıklı kısmı iki haneli girilen pozitif bir rasyonel sayının ondalıklı kısmının sayı değerinin bir tam kare olup olmadığını araştıran algoritma ve akış şemasının oluşturulması.

Örnek : Ondalıklı kısmı iki haneli girilen pozitif bir rasyonel sayının ondalıklı kısmının sayı değerinin bir tam kare olup olmadığını araştıran algoritma ve akış şemasının oluşturulması.

A1. Başla,
A2. A sayısını gir,
A3. B=A-TAM(A) ,
A4. DEGER=B*100 ,
A5. SAYI=(TAM(DEGER^(1/2)))^2 ,
A6. Eğer SAYI=DEGER ise A8 ’e git,
A7. “Bu sayı tam kare değil” yaz ve A9 ’a git,
A8. “Bu sayı tam karedir” yaz,
A9. Dur.

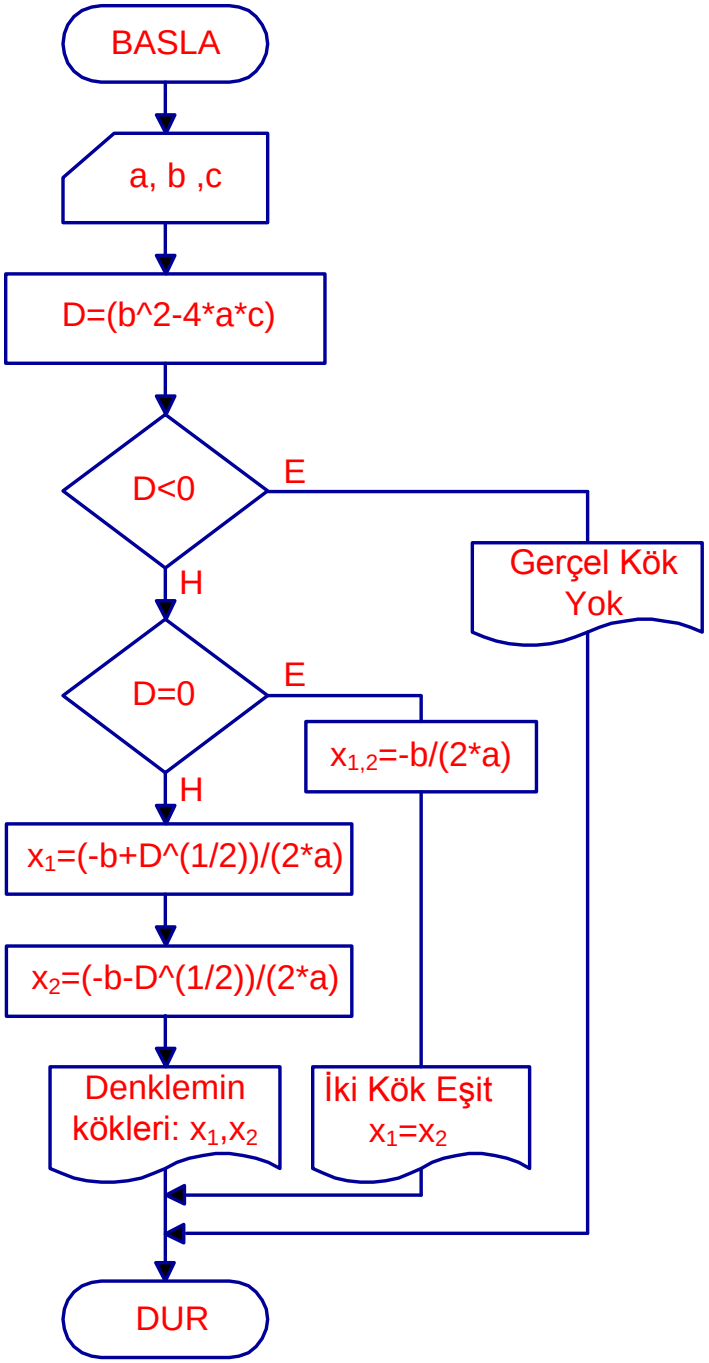


Matematiksel ifadelerin bilgisayar dilinde kodlanması

Matematiksel yazım	**Bilgisayara kodlanması**
$a + b - c + 2abc - 7$	a+b-c+2*a*b*c-7
$a + b^2 - c^3$	a+sqr(b)-pow(c,3)
$a - \frac{b}{c} + 2ac - \frac{2}{a+b}$	a-b/c+2*a*c-2/(a+b)
$\sqrt{a+b} - \frac{2ab}{b^2 - 4ac}$	sqrt(a+b)-(2*a*b)/(sqr(b)-4*a*c)
$\frac{a+b-c}{\sqrt{a^2+b^3}} - \frac{2(ab+ac+bc)}{9}$	(a+b-c)/sqrt(sqr(a)+pow(b,3))-(2(a*b+a*c+b*c))/9
$a + \frac{b + c^2}{d + \frac{e-f}{3a}}$	a+(b+sqr(2))/(d+((e-f)/(3*a)))
$\frac{a^5 + \frac{b^3}{c+d}}{\sqrt[3]{a + b^2 + c^3 + d^4}}$	(pow(a,5)+(pow(b,3)/(c+d)))/sqrt(a+sqr(b)+pow(c,3)+pow(d,4))

Örnek : İkinci dereceden denklemin köklerini bulan programın akış şeması

Örnek : İkinci dereceden denklemin köklerini bulan programın akış şeması



Örnek : İstenildiği kadar elemandan oluşan bir sayı dizisinde negatif ve pozitif elemanların sayısını bulan algoritma ve akış şemasının oluşturulması.

Örnek : İstenildiği kadar elemandan oluşan bir sayı dizisinde negatif ve pozitif elemanların sayısını bulan algoritma ve akış şemasının oluşturulması.

A1. Başta,
A2. I=1, N=0,P=0 ,
A3. A(I) ’yı gir,
A4. Eğer A(I)=0 ise A7. adıma git,
A5. Eğer A(I)<0 ise N=N+I ve A7. adıma git,
A6. P=P+I ,
A7. C ’yi gir (Devam Etmek İstiyormusunuz:E/H?)
A8. Eğer C ”H” ise A10. adıma git,
A9. I=I+1 ve A3. adıma geri dön,
A10. P ’yi ve N ’yi yaz,
A11. Dur.