MATLAB'te PROGRAMLAMA

Bir bilgisayar programı, bir dizi bilgisayar komutudur. Basit bir programda, komutlar yazıldıkları sırada birbiri ardına yürütülür. Bu kitapta, örneğin, MATLAB komut dosyalarında şimdiye kadar sunulan tüm programlar basit programlardır. Bununla birlikte, birçok durum, komutların mutlaka yazıldıkları sırayla çalıştırılmadığı veya program farklı girdi değişkenleriyle çalıştığında farklı komutların (veya komut gruplarının) çalıştırıldığı daha karmaşık programlar gerektirir. Örneğin, bir paketi postalama maliyetini hesaplayan bir bilgisayar programı, paketin ağırlığına ve boyutuna, içeriğine (kitapların postalanması daha ucuzdur) ve hizmet türüne (havayolu postası, e-posta) bağlı olarak maliyeti hesaplamak için farklı matematiksel ifadeler kullanır. Diğer durumlarda, bir program içinde bir dizi komutu birkaç kez tekrar etme ihtiyacı olabilir. Örneğin, denklemleri çözen programlar, cevaptaki hata bir ölçüden daha küçük olana kadar bir dizi hesaplamayı tekrar eder.

MATLAB, bir programın akışını kontrol etmek için kullanılabilecek çeşitli araçlar sağlar. Koşul deyimleri ve anahtar (switch) yapısı, komutları atlamayı veya farklı durumlarda belirli komut gruplarını çalıştırmayı mümkün kılar. For Döngüleri ve while döngüleri, bir komut dizisini birkaç kez tekrar etmeyi mümkün kılar.

Bir programın akışını değiştirmenin program içinde bir tür karar verme süreci gerektirdiği açıktır. Bilgisayar, sonraki komutu yürütmeye mi yoksa bir veya daha fazla komutu atlayıp programda farklı bir satırda devam etmeye mi karar vermelidir. Program bu kararları değişkenlerin değerlerini karşılaştırarak verir. Bu, Bölümde açıklanan ilişkisel ve mantıksal operatörler kullanılarak yapılır.

Kullanıcı tanımlı işlevlerin (İlerleyen Bölümlerde anlatılacak) programlamada kullanılabileceği de unutulmamalıdır. Kullanıcı tanımlı bir işlev, bir alt program olarak kullanılabilir. Ana program, kullanıcı tanımlı işleve sahip komut satırına ulaştığında, işleve girdi sağlar ve sonuçları "bekler". Kullanıcı tanımlı işlev, hesaplamaları gerçekleştirir ve sonuçları ana programa geri aktarır ve ardından bir sonraki komuta geçer.

İLİŞKİLİ VE MANTIK OPERATÖRLER

İlişkisel bir operatör, bir karşılaştırma ifadesinin (örneğin, 5 <8) doğru mu yanlış mı olduğunu belirleyerek iki sayıyı karşılaştırır. İfade doğruysa, 1 değeri atanır. İfade yanlışsa, 0 değeri atanır. Mantıksal bir operatör doğru / yanlış ifadelerini inceler ve doğru (1) veya yanlış (0) bir sonuç üretir belirli operatöre göre. Örneğin, mantıksal AND operatörü, yalnızca her iki ifade de doğruysa 1 verir. İlişkisel ve mantıksal operatörler matematiksel ifadelerde kullanılabilir ve bu bölümde gösterileceği gibi, bir bilgisayar programının akışını kontrol eden kararlar almak için diğer komutlarla birlikte sıklıkla kullanılır.

İlişkisel operatörler:

MATLAB'deki ilişkisel operatörler şunlardır:

İlişkisel operatör	Açıklama
<	Küçüktür
>	Büyüktür
<=	Küçüktür veya eşittir
> =	Büyük veya eşit
= =	Eşittir
~ =	Eşit Değil

Bir = işareti atama operatörü olduğundan "eşittir" ilişkisel operatörün iki = işaretinden (aralarında boşluk olmayan) oluştuğuna dikkat edin. İki karakterden oluşan diğer ilişkisel operatörlerde, karakterler arasında boşluk da yoktur (<=,> =, ~ =).

- İlişkisel operatörler, matematiksel bir ifadede aritmetik operatörler olarak kullanılır. Sonuç, diğer matematiksel işlemlerde, dizileri adreslemede ve bir programın akışını kontrol etmek için diğer MATLAB komutlarıyla (örneğin eğer) birlikte kullanılabilir.
- İki sayı karşılaştırıldığında, ilişkisel operatöre göre karşılaştırma doğruysa sonuç 1 (mantıksal doğru) ve karşılaştırma yanlışsa 0 (mantıksal yanlış) olur.
- İki skaler karşılaştırılırsa, sonuç skaler 1 veya 0 olur. İki dizi karşılaştırılırsa (yalnızca aynı boyuttaki diziler karşılaştırılabilir), karşılaştırma öğe öğe yapılır ve sonuç, mantıksal bir dizidir ve her adresteki karşılaştırma sonucuna göre 1'ler ve 0'lar ile aynı boyuttur.
- Bir skaler bir dizi ile karşılaştırılırsa, skaler, dizinin her elemanıyla karşılaştırılır ve sonuç, her bir elemanın karşılaştırmasının sonucuna göre 1'ler ve 0'lar içeren mantıksal bir dizidir.

Örnek

```
>> 5>8
ans =
0
>> a=5<10
a =
1
>> y=(6<10)+(7>8)+(5*3= =60/4)
y =
2
>> b=[15 6 9 4 11 7 14]; c=[8 20 9 2 19 7 10];
>> d=c>=b
d =
0 1 1 0 1 1 0
>> b == c
ans =
0 0 1 0 0 1 0
>> b~=c
ans =
```

```
1 1 0 1 1 0 1
>> f=b-c>0
f =
1 0 0 1 0 0 1
>> A=[2 9 4; -3 5 2; 6 7 -1]
A =
2 9 4
-3 5 2
6 7 -1
>> B=A<=2
B =
1 0 0
1 0 1
0 0 1
```

0'lar ve 1'ler olan vektörler olan vektörlerle ilişkisel bir işlemin sonuçlarına mantıksal vektörler denir ve vektörleri adreslemek için kullanılabilir. Başka bir vektörü adreslemek için mantıksal bir vektör kullanıldığında, bu vektörden mantıksal vektörün 1'lere sahip olduğu konumlardaki öğeleri çıkarır.
Örneğin:

```
>> r = [8 12 9 4 23 19 10]
r =
8 12 9 4 23 19 10
>> s=r<=10
s =
1 0 1 1 0 0 1
>> t=r(s)
t =
8 9 4 10
>> w=r(r<=10)
w =
8 9 4 10
```

Sayıları 0 ve 1 olan sayısal vektörler ve diziler, mantıksal vektörler ve 0 ve 1'li dizilerle aynı değildir. Sayısal vektörler ve diziler adresleme için kullanılamaz. Mantıksal vektörler ve diziler ise aritmetik işlemlerde kullanılabilir. Aritmetik işlemlerde ilk kez bir mantıksal vektör veya dizi kullanıldığında, sayısal bir vektör veya diziye değiştirilir.

Öncelik sırası: İlişkisel ve aritmetik işlemleri içeren matematiksel bir ifadede, aritmetik işlemler (+, -, *, /, \) ilişkisel işlemlere göre önceliğe sahiptir. İlişkisel operatörlerin kendileri eşit önceliğe sahiptir ve soldan sağa doğru değerlendirilir. Öncelik sırasını değiştirmek için parantezler kullanılabilir. Örnekler:

```
>> 3+4<16/2
ans =
1
>> 3+(4<16)/2
ans =
3.5000
```

Mantıksal operatör	Adı	Açıklama
& A&B	Ve (AND)	İki işlenenle çalışır (A ve B). Eğer ikisi de doğru ise, sonuç doğrudur (1); aksi takdirde sonuç yanlıştır (0).
| A|B	Veya (OR)	İki işlenen üzerinde çalışır (A ve B). Eğer biri veya ikisi de doğru ise, sonuç doğru (1); aksi takdirde (ikisi de yanlış ise) sonuç yanlıştır (0).
~ ~A	Değil (NOT)	Bir işlenen (A) üzerinde çalışır. İşlenenin tersini verir; işlenen yanlışsadoğru (1) ve işlenen doğruysayanlış (0).

- Mantıksal operatörler, işlenenler olarak sayılara sahiptir. Sıfır olmayan bir sayı doğrudur ve sıfır sayı yanlıştır.
- Mantıksal operatörler (ilişkisel operatörler gibi) matematiksel bir ifadede aritmetik operatörler olarak kullanılır. Sonuç, diğer matematiksel işlemlerde, dizileri adreslemede ve bir programın akışını kontrol etmek için diğer MATLAB komutlarıyla (örneğin eğer) birlikte kullanılabilir.
- Mantıksal operatörler (ilişkisel operatörler gibi) skalerler ve dizilerle kullanılabilir.
- Mantıksal işlemler AND ve OR, her iki işlenen de skaler olarak olabilir, hem diziler olarak, hem de bir dizi ve diğeri de skaler olarak. Her ikisi de skaler ise, sonuç skaler 0 veya 1'dir. Eğer her ikisi de diziyse, aynı boyutta olmalıdır ve mantıksal işlem eleman-eleman yapılır. Sonuç, her konumdaki işlemin sonucuna göre 1'ler ve 0'larla aynı boyutta bir dizidir. Bir işlenen bir skaler ve diğeri bir diziyse, mantıksal işlem skaler ile dizideki elemanların her biri arasında yapılır ve sonuç, 1'ler ve 0'lar ile aynı boyutta bir dizidir.
- Mantıksal işlemin bir işlenen YOKTUR. Bir skaler ile kullanıldığında, sonuç skaler 0 veya 1'dir. Bir dizi ile kullanıldığında, sonuç, dizinin sıfır olmayan sayılara sahip olduğu konumlarda 0'lı ve aynı boyutta bir dizidir. dizi 0'lara sahiptir.

Örnekler:

```
>> 3&7
ans =
1
```

3 ve 7'nin ikisi de doğrudur (sıfırdan farklıdır), dolayısıyla sonuç 1'dir.

```
>> a=5|0
a =
1
```

En az bir sayı doğru olduğu için (sıfır olmayan) 1'e atanır.

>> ~25
ans =
0
25 doğru (sıfır olmayan) ve tersi yanlış olduğu için sonuç 0'dır
>> t=25*((12&0)+(~0)+(0|5))
t =
50

>> x=[9 3 0 11 0 15]; y=[2 0 13 -11 0 4];
>> x&y
ans =
1 0 0 1 0 1
Sonuç, hem x hem de y'nin doğru olduğu (sıfır olmayan öğeler) her konumda 1 ve aksi takdirde 0 lar olan bir vektördür.
>> z=x|y
z =
1 1 1 1 0 1

Sonuç, x ve y'den birinin veya her ikisinin de doğru olduğu (sıfır olmayan öğeler) her konumda 1 ve aksi takdirde 0s olan bir vektördür.

~(x+y)
ans =
0 0 0 1 1 0

Sonuç, x + y vektörünün doğru olduğu her konumda 0 (sıfır olmayan öğeler) ve x + y'nin yanlış olduğu her konumda 1 (sıfır öğeler) olan bir vektördür.

Öncelik sırası:

Aritmetik, ilişkisel ve mantıksal operatörler matematiksel ifadelerde birleştirilebilir. Bir ifade böyle bir kombinasyona sahip olduğunda, sonuç, işlemlerin gerçekleştirildiği sıraya bağlıdır. Aşağıdaki MATLAB tarafından kullanılan sıra:

Öncelik	İşlemi
1	(en yüksek) Parantezler (iç içe parantezler varsa, içteki parantezler öncelik)
2	Üs alma
3	Mantıksal DEĞİL (~)
4	Çarpma, bölme
5	Toplama, çıkarma
6	İlişkisel operatörler (>, <,> =, <=, = =, ~ =)
7	Mantıksal AND (&)
8	(en düşük) Mantıksal VEYA (|)

İki veya daha fazla işlem aynı önceliğe sahipse, ifade soldan sağa sırayla yürütülür.

Burada, yukarıda gösterilen sıranın MATLAB 6'dan beri kullanıldığına dikkat edilmelidir. MATLAB'ın önceki sürümleri biraz farklı bir sıra kullanıyordu (& önceliği yoktu), bu yüzden kullanıcı dikkatli olmalı. MATLAB'ın farklı sürümleri arasındaki uyumluluk sorunları, gerekmediğinde bile parantez kullanılarak önlenebilir.

Örnek:

>> x=-2; y=5;
>> -5<x<-1
ans =
0
Bu eşitsizlik matematiksel olarak doğrudur. MATLAB soldan sağa çalıştığı için cevap yanlıştır. –5 <x doğrudur (= 1) ve sonra 1 <–1 yanlıştır (0).

>> -5<x & x<-1
ans =
1
Matematiksel olarak doğru ifade, & mantıksal operatörü kullanılarak elde edilir. Eşitsizlikler önce hesaplanır. Her ikisi de doğru (1) olduğundan, cevap 1'dir.
>> ~(y<7)
ans =
0
Önce y <7 çalıştırılır, bu doğrudur (1) ve ~ 1, 0'dır.
>> ~y<7

ans =
1
~ y önce çalıştırılır, y doğrudur (1) (y sıfır olmadığı için), ~ 1 0 ve 0 <7 doğrudur (1).
>> ~((y>=8)|(x<-1))
ans =
0
önce y> = 8 (yanlış) ve x <–1 (doğru) çalıştırılır. VEYA sonraki çalıştırılır (doğru). ~ en son çalıştırılır ve yanlış (0) verir.
>> ~(y>=8)|(x<-1)
ans =
1

önce y> = 8 (yanlış) ve x <–1 (doğru) çalıştırılır. Sonra (y> = 8) DEĞİL yürütülür (doğru). VEYA en son çalıştırılır ve doğru (1) verir.

Yerleşik mantıksal işlevler:

MATLAB, mantıksal işleçlere eşdeğer yerleşik işlevlere sahiptir. Bu işlevler şunlardır:

`and(A, B)` `A & B` 'ye eşdeğerdir

`or (A, B)` `A | B` 'ye eşdeğer

`not(A)` `~ A` ile eşdeğer

Ek olarak, MATLAB, bazıları aşağıdaki tabloda açıklanan başka mantıksal yerleşik işlevlere sahiptir:

Fonksiyon	Açıklaması	Örnek
`xor(a,b)`	Özel veya. Bir işlenen doğru ve diğeri yanlışsa doğruya (1) döndürür.	>> xor(7,0) ans = 1 >> xor(7,-5) ans = 0
`all(A)`	A vektöründeki tüm öğeler doğruysa (sıfır olmayan) 1 (doğru) döndürür. Bir veya daha fazla öğe yanlışsa 0 (yanlış) döndürür (sıfır). A bir matris ise, A'nın sütunlarını vektör olarak ele alır ve 1'li ve 0'lı bir vektör döndürür.	>> A=[6 2 15 9 7 11]; >> all(A) ans = 1 >> B=[6 2 15 9 0 11]; >> all(B) ans = 0
`any(A)`	A vektöründeki herhangi bir öğe doğruysa (sıfır olmayan) 1 (doğru) döndürür. Tüm öğeler yanlışsa (sıfır) 0 (yanlış) döndürür. A bir matris ise, A'nın sütunlarını vektör olarak ele alır ve 1s ve 0'lı bir vektör döndürür.	>> A=[6 0 15 0 0 11]; >> any(A) ans = 1 >> B = [0 0 0 0 0 0]; >> any(B) ans = 0

`find(A)`
 	
`find(A>d)`
	A bir vektör ise, sıfır olmayan öğelerin indislerini döndürür.
	A bir vektörse, d'den büyük olan öğelerin adresini döndürür (herhangi bir ilişkisel operatör kullanılabilir).
		>> A=[0 9 4 3 7 0 0 1 8];
		>> find(A)
		ans =
		2 3 4
		5 8 9
		>> find(A>4)
		ans =
		2 5 9

GİRDİ		**ÇIKTI**				
A	**B**	**AND (A&B)**	**OR (A|B)**	**XOR (A,B)**	**NOT ~ A**	**NOT ~B**
Yanlış	**Yanlış**	**Yanlış**	**Yanlış**	**Yanlış**	**Doğru**	**Doğru**
Yanlış	**Doğru**	**Yanlış**	**Doğru**	**Doğru**	**Doğru**	**Yanlış**
Doğru	**Yanlış**	**Yanlış**	**Doğru**	**Doğru**	**Yanlış**	**Doğru**
Doğru	**Doğru**	**Doğru**	**Doğru**	**Yanlış**	**Yanlış**	**Yanlış**

Alıştırma

Aşağıdakiler, 2002 Nisan ayı boyunca Washington, DC'deki günlük maksimum sıcaklıklardır (° F cinsinden): 58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89 91 80 59 69 56 64 63 66 64 74 63 69 (ABD Ulusal Okyanus ve Atmosfer İdaresi verileri). Aşağıdakileri belirlemek için ilişkisel ve mantıksal işlemleri kullanın:

(a) Sıcaklığın 75 ° 'nin üzerinde olduğu gün sayısı.

(b) Sıcaklığın 65 ° ile 80 ° arasında olduğu gün sayısı.

(c) Sıcaklığın 50 ° ile 60 ° arasında olduğu ayın günleri.

Çözüm:

Komut dosyasında aşağıdaki sıcaklıklar bir vektöre girilir. İlişkisel ve mantıksal ifadeler daha sonra verileri analiz etmek için kullanılır.

```
T=[58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 ...
91 93 89 91 80 59 69 56 64 63 66 64 74 63 69];
```

Tabove75=T>=75; % T> = 75 olan adreslerde 1'li bir vektör.

NdaysTabove75=sum(Tabove75) % Tabove75 vektöründeki tüm 1'leri toplayın.

Tbetween65and80=(T>=65)&(T<=80) %; T> = 65 ve T <= 80 olan adreslerde 1'li bir vektör.

NdaysTbetween65and80=sum(Tbetween65and80) % Tbetween65and80 vektöründeki tüm 1'leri toplayın

datesTbetween50and60=find((T>=50)&(T<=60)) % Find işlevi, T'deki 50 ile 60 arasında değerlere sahip öğelerin adresini döndürür.

Sonuç ekranında

```
>> Exp6_1
NdaysTabove75 =
7
NdaysTbetween65and80 =
12
datesTbetween50and60 =
1 4 5 7 21 23
```

KOŞULLU İFADELER

Koşullu ifade, MATLAB'ın koşullu ifadeyi izleyen bir grup komutun çalıştırılıp çalıştırılmayacağına veya bu komutların atlanıp atlanmayacağına karar vermesine izin veren bir komuttur. Koşullu bir ifadede, koşullu bir ifade belirtilir. İfade doğruysa, ifadeyi izleyen bir grup komut yürütülür. İfade yanlışsa, bilgisayar grubu atlar. Koşullu ifadenin temel biçimi şudur:

if ilişkisel ve / veya mantıksal operatörlerden oluşan koşullu ifade

Örnekler: Tüm değişkenlerin atanmış değerleri olmak üzere;

if a<b

if c>=5

if a==b

if a~=0

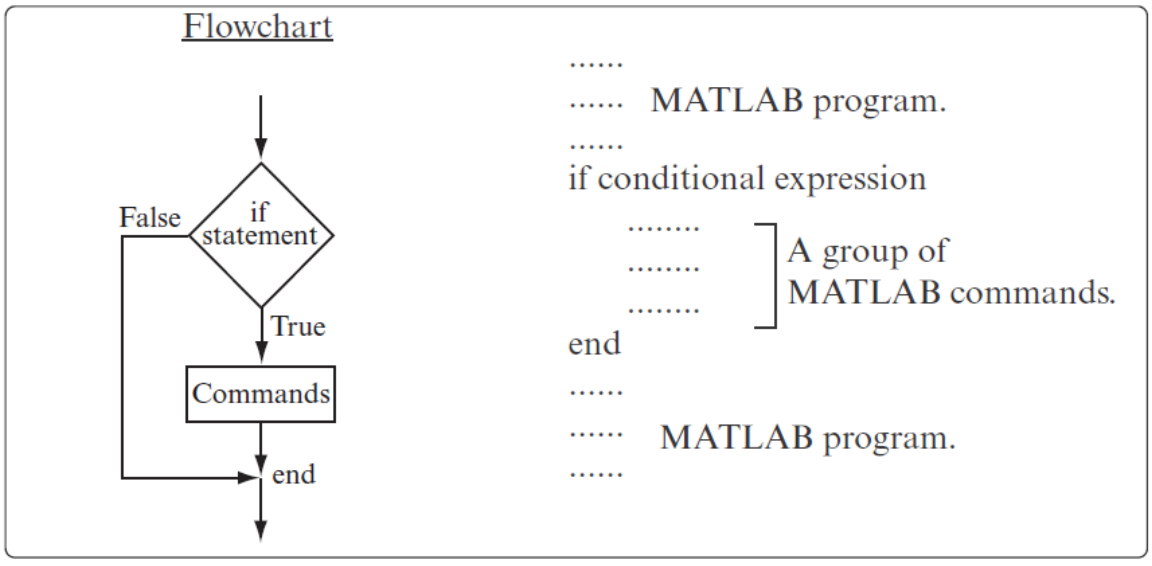
if (d<h)&(x>7)

if (x~=13)|(y<0)

- Koşullu ifadeler, bir komut dosyası içinde yazılmış bir programın parçası veya kullanıcı tanımlı bir işlev olabilir (Bölüm 7).
- Aşağıda gösterildiği gibi, her if ifadesi için bir bitiş ifadesi vardır.
- İf ifadesi genellikle üç yapıda kullanılır, if-end, if-else-end ve if-elseif-else-end, aşağıda açıklanacaktır.

İf-end Yapısı:

İf-end koşullu ifadesi Şekilde şematik olarak gösterilmektedir. Şekil, komutların programda nasıl yazıldığını ve akışı veya komutların yürütüldüğü sırayı sembolik olarak gösteren bir akış şemasını gösterir. Program çalışırken if ifadesine ulaşır. İf ifadesindeki koşullu ifade true (1) ise, program, if ifadesini sonuna kadar takip eden komutları yürütmeye devam eder. Koşullu ifade yanlış (0) ise, program if ve son arasındaki komut grubunu atlar ve sondan sonra gelen komutlarla devam eder.



Ekranda mavi renkte if ve end sözcükleri görünür ve komutlar if ifadesi ile end ifadesi arasında otomatik olarak girintilenir (olması gerekmez), bu da programın okunmasını

kolaylaştırır. Bir komut dosyasında if-end ifadesinin kullanıldığı bir örnek Örnek Problem'de gösterilmiştir

Örnek Problem: İşçinin ücretini hesaplama

Bir işçiye 40 saate kadar saatlik ücretine göre, fazla mesai için% 50 daha fazla ödeme yapılır. Bir işçinin ücretini hesaplayan bir MATLAB dosyasına bir program yazın. Program, kullanıcıdan saat sayısını ve saatlik ücreti girmesini ister. Program daha sonra ödemeyi görüntüler.

Çözüm

Bir MATLAB dosyasındaki program aşağıda gösterilmiştir. Program ilk olarak saat sayısını saat ücreti ile çarparak ücreti hesaplar. Daha sonra bir if ifadesi, saat sayısının 40'tan fazla olup olmadığını kontrol eder. Öyleyse, sonraki satır çalıştırılır ve 40'ın üzerindeki saatler için ekstra ödeme eklenir. Değilse, program sona atlar.

```
t=input('Lütfen isçinin calistigi saati sayisini giriniz ');
h=input(' Lütfen saatlik ücreti TL olarak girin ');
Maas=t*h;
if t>40

Maas = Maas +(t-40)*0.5*h;
end
fprintf('iscinin maası %5.2f TL ', Maas)
```

Programın iki durum için uygulanması (Komut Penceresinde) aşağıda gösterilmiştir (dosya maasT olarak kaydedilmiştir):

Lutfen isçinin calistigi saati sayisini giriniz 25

Lutfen saatlik ücreti TL olarak girin 50

iscinin maasi 1250.00 TL>>

Lutfen isçinin calistigi saati sayisini giriniz 45

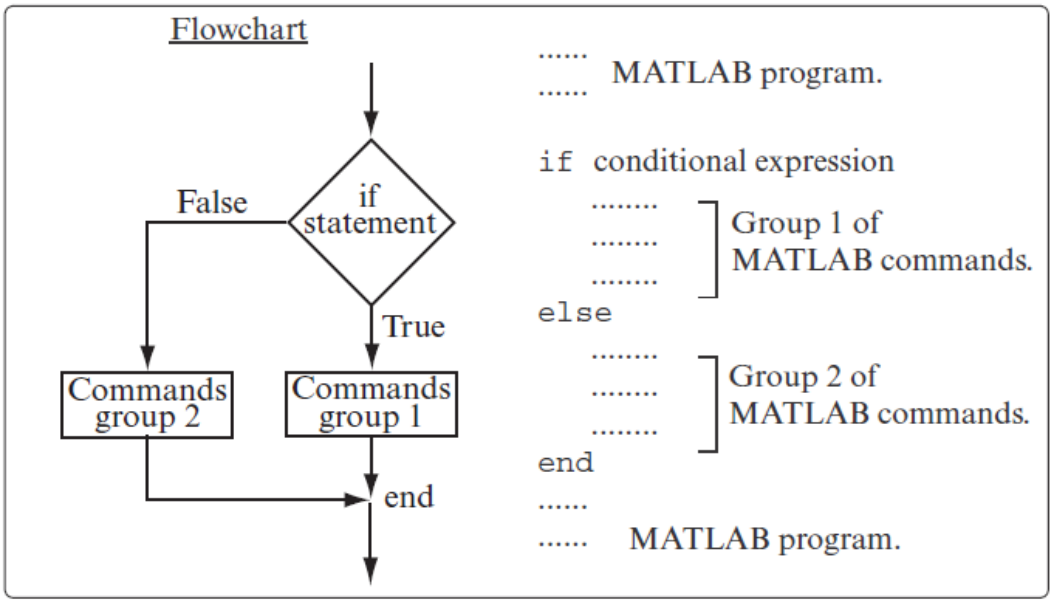
Lutfen saatlik ücreti TL olarak girin 50

iscinin maasi 2375.00 TL>>

İf-else-end Yapısı:

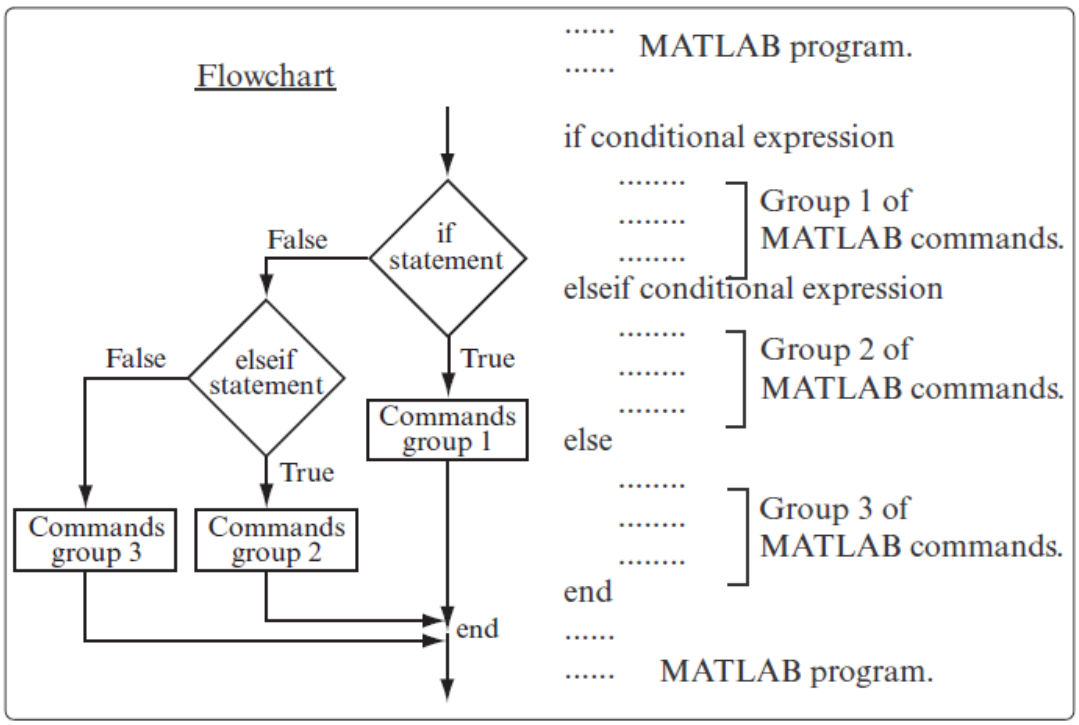
İf-else-end yapısı, yürütme için olası iki gruptan bir komut grubunu seçmek için bir araç sağlar. If-else-end yapısı Aşağıdaki şekilde gösterilmektedir. Şekil, komutların programda nasıl yazıldığını gösterir ve komutların yürütüldüğü akışı veya sırayı gösteren bir akış şeması içerir. İlk satır, koşullu ifadeye sahip bir if ifadesidir. Koşullu ifade doğruysa, program if ve else ifadeleri arasındaki grup 1 komutlarını çalıştırır ve ardından sona atlar. Koşullu ifade

anlışsa, program else'e atlar ve sonra else ile son arasında grup 2 komutunu yürütür.



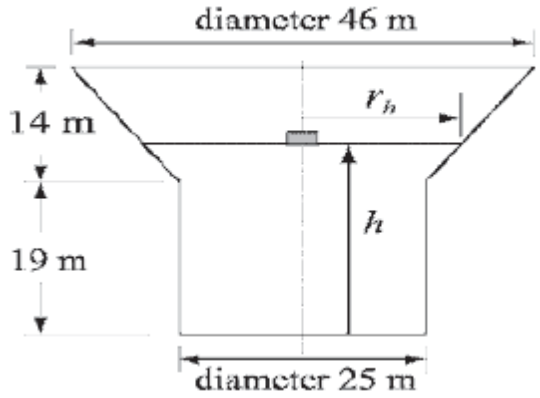
İf-elseif-else-end Yapısı

İf-elseif-else-end yapısı Aşağıdaki Şekilde gösterilmektedir. Şekil, komutların programda nasıl yazıldıklarını gösterir ve komutların yürütüldüğü akışı veya sırayı gösteren bir akış şeması verir. Bu yapı, yürütme için üç komut grubundan birini seçmeyi mümkün kılan iki koşullu ifade (if ve elseif) içerir. İlk satır, koşullu ifadeye sahip bir if ifadesidir. Koşullu ifade doğruysa, program if ve elseif ifadeleri arasındaki grup 1 komutlarını çalıştırır ve ardından sona atlar. İf ifadesindeki koşullu ifade yanlışsa, program elseif ifadesine atlar. Elseif ifadesindeki koşullu ifade doğruysa, program elseif ve else arasındaki grup 2 komutunu yürütür ve sonra sona atlar. Elseif ifadesindeki koşullu ifade yanlışsa, program else'e atlar ve else ile son arasında 3. grup komutu yürütür.



Burada birkaç elseif ifadesinin ve ilişkili komut gruplarının eklenebileceği belirtilmelidir. Bu şekilde daha fazla koşul dahil edilebilir. Ayrıca, else ifadesi isteğe bağlıdır. Bu, birkaç elseif deyimi ve başka bir deyim olmaması durumunda, koşullu ifadelerden herhangi biri doğruysa, ilişkili komutların çalıştırılacağı anlamına gelir; aksi takdirde hiçbir şey yürütülmez.

Örnek Problem: Su kulesindeki su seviyesi



Bir su kulesindeki tank, şekilde gösterilen geometriye sahiptir (alt kısım bir silindirdir ve üst kısım bir koninin ters çevrilmiş bir kesik halidir). Tankın içinde su seviyesini gösteren bir şamandıra bulunmaktadır. Şamandıranın konumundan (h yüksekliği) tanktaki suyun hacmini belirleyen bir MATLAB programı yazınız. Program, kullanıcıdan m cinsinden bir h değeri girmesini ister ve çıktı olarak suyun hacmini m^3 cinsinden görüntüler.

For $0 \le h \le 19$ m the volume of the water is given by the volume of a cylinder with height h: $V = \pi 12.5^2 h$.
For $19 \le h \le 33$ m the volume of the water is given by adding the volume of a cylinder with $h = 19$ m, and the volume of the water in the cone:

$$V = \pi 12.5^2 \cdot 19 + \frac{1}{3}\pi(h-19)(12.5^2 + 12.5 r_h + r_h^2)$$

where $r_h = 12.5 + \frac{10.5}{14}(h-19)$.

The program is:

```
% The program calculates the volume of the water in the
water tower.
h=input('Please enter the height of the float in meter  ');
if h > 33
   disp('ERROR. The height cannot be larger than 33 m.')
elseif h < 0
   disp('ERROR. The height cannot be a negative number.')
elseif h <= 19
   v = pi*12.5^2*h;
   fprintf('The volume of the water is %7.3f cubic meter.\n',v)
```

```
else
   rh=12.5+10.5*(h-19)/14;
   v=pi*12.5^2*19+pi*(h-19)*(12.5^2+12.5*rh+rh^2)/3;
   fprintf('The volume of the water is %7.3f cubic meter.\n',v)
end
```

The following is the display in the Command Window when the program is used with three different values of water height.

```
Please enter the height of the float in meter  8
The volume of the water is 3926.991 cubic meter.
Please enter the height of the float in meter  25.7
The volume of the water is 14114.742 cubic meter.
Please enter the height of the float in meter  35
ERROR. The height cannot be larger than 33 m.
```

switch-case yapısı:

Switch-case ifadesi, bir programın akışını yönlendirmek için kullanılabilecek başka bir yöntemdir. Birkaç olası gruptan yürütülmesi için bir komut grubu seçmek için bir yol sağlar. İfadenin yapısı Şekilde gösterilmektedir. İlk satır, şu forma sahip olan anahtar komutudur:

switch switch ifadesi

Switch ifadesi bir skaler veya bir dize olabilir. Genellikle, atanmış bir skaler veya dizeye sahip bir değişkendir. Bununla birlikte, önceden atanmış değişkenleri içeren ve değerlendirilebilen matematiksel bir ifade de olabilir.

- Geçiş komutlarının ardından bir veya birkaç durum komutu vardır. Her birinin yanında bir değer (skaler veya dize olabilir) (değer1, değer2, vb.) Ve altında ilişkili bir komut grubu vardır.
- Son durum komutundan sonra, isteğe bağlı başka bir komut ve ardından bir grup komut vardır.
- Son satırı end ile bitmelidir.

```
.....
.....      MATLAB program.

switch switch expression
    case value1
    ........
    ........  ]  Group 1 of commands.
    case value2
    ........
    ........  ]  Group 2 of commands.
    case value3
    ........
    ........  ]  Group 3 of commands.
    otherwise
    ........
    ........  ]  Group 4 of commands.
end
.....
.....   MATLAB program.
```

Switch-case ifadesi nasıl çalışır?

Anahtar komutundaki switch ifadesinin değeri, her bir case ifadesinin yanındaki değerlerle karşılaştırılır. Bir eşleşme bulunursa, eşleşme ile case ifadesini izleyen komut grubu çalıştırılır. (Yalnızca bir komut grubu - birbiriyle eşleşen durum ile durum arasında olanı, aksi takdirde veya sonraki son deyimi çalıştırılır).

Birden fazla eşleşme varsa, yalnızca ilk eşleşen durum yürütülür.

- Eşleşme bulunmazsa ve otherwise aksi takdirde (isteğe bağlı olan) ifadesi mevcutsa, aksi ve son arasındaki komutlar grubu çalıştırılır.
- Eşleşme bulunmazsa ve otherwise ifade mevcut değilse, komut gruplarının hiçbiri çalıştırılmaz.
- Bir durum ifadesi birden fazla değere sahip olabilir. Bu, değerleri şu biçimde yazarak yapılır: {değer1, değer2, değer3, ...}. (Bu kitapta ele alınmayan bu forma hücre dizisi adı verilir.) Durum, değerlerden en az biri anahtar ifadesinin değeriyle eşleşirse yürütülür.

Not: *MATLAB'da yalnızca ilk eşleşen durum yürütülür. Eşleşen ilk durumla ilişkili komut grubu çalıştırıldıktan sonra, program bitiş ifadesine geçer. Bu, break ifadelerinin gerekli olduğu C dilinden farklıdır.*

Örnek Problem Enerji birimlerini dönüştürme

Joule, ft-lb, cal veya eV birimlerinde verilen enerji miktarını (iş), kullanıcı tarafından belirtilen farklı birimlerdeki eşdeğer miktara dönüştüren bir program yazın. Program kullanıcıdan enerji miktarını, mevcut birimlerini ve istenen yeni birimleri girmesini ister. Çıktı, yeni birimlerdeki enerji miktarıdır.

Dönüşüm faktörleri şunlardır: $1\text{J} = 0.738\ \text{ft} - \text{Ib} = 0.239\text{cal} = 6{,}24\text{x}10^{18}\text{eV}$.

Programı kullanarak

a)150 J ü ft-lb ye

b) 2800 cal yi J ye

c)2.7 eV u cal birimine dönüştürünüz

Program iki set switch-case deyimi ve bir if-else-end deyimi içerir. İlk switch-case ifadesi, girdi miktarını başlangıç birimlerinden joule birimlerine dönüştürmek için kullanılır. İkincisi, miktarı joule'den belirtilen yeni birimlere dönüştürmek için kullanılır. İf-else-end ifadesi, birimler yanlış girilirse bir hata mesajı oluşturmak için kullanılır.

```
Ein=input('Donusturulecek enerjinin (is) degerini girin: ');
EinUnits=input('Mevcut birimi girin (J, ft-lb, cal, veya eV): ','s');
EoutUnits=input('Yeni birimi girin (J, ft-lb, cal, or eV): ','s');

error=0;   % Değişken hataya 0 atayın.
switch EinUnits    % İlk switch ifadesi. Switch ifadesi, başlangıç birimleri olan bir dizedir.
case 'J'
EJ=Ein;
case 'ft-lb'
EJ=Ein/0.738;
case 'cal'
EJ=Ein/0.239;
case 'eV'
EJ=Ein/6.24e18;
% Dört durum ifadesinin her biri, ilk birimlerden birine karşılık gelen bir değere (dize) ve Ein'i J
%birimlerine dönüştüren bir komuta sahiptir (Değeri EJ'ye atayın.)
otherwise
error=1; %Eşleşme bulunmazsa hatayı 1'e atayın. Yalnızca başlangıç birimleri yanlış yazıldığında
%mümkündür.
end

switch EoutUnits    % ikinci Switch ifadesi, yeni birimleri olan bir dizedir.
case 'J'
Eout=EJ;
case 'ft-lb'
Eout=EJ*0.738;
case 'cal'
Eout=EJ*0.239;
case 'eV'
```

```
Eout=EJ*6.24e18;

%Dört case deyiminin her biri, yeni birimlerden birine karşılık gelen bir değere (dize) ve EJ'yi yeni
%birimlere dönüştüren bir komuta sahiptir. (Değeri Eout'a atayın.)

otherwise
error=1;
end
if error % If-else-end yapısıt.
disp('ERROR mevcut vaya yeni birimler yanlis yazilmistir ')  % Hata doğruysa (sıfır değilse), bir
hata %mesajı görüntüleyin.
else
fprintf('E = %g %s',Eout,EoutUnits)  % Hata yanlışsa (sıfır), dönüştürülen enerjiyi görüntüleyin
end
```

Örnek olarak, (EnergyConversion olarak kaydedilen) komut dosyası dosyası, problem ifadesinin (b) bölümünde dönüştürme yapmak için Komut Penceresinde daha sonra kullanılır.

```
>> EnergyConversion


Donusturulecek enerjinin (is) degerini girin: 4800

Mevcut birimi girin (J, ft-lb, cal, veya eV): cal

Yeni birimi girin (J, ft-lb, cal, or eV): J

E = 20083.7 J>>
```

DÖNGÜLER

Döngü, bir bilgisayar programının akışını değiştirmenin başka bir yöntemidir. Bir döngüde, bir komutun veya bir grup komutun yürütülmesi arka arkaya birkaç kez tekrarlanır. Her yürütme turuna geçiş denir. Her geçişte en az bir değişken, ancak genellikle birden fazla, hatta döngü içinde tanımlanan tüm değişkenlere yeni değerler atanır. MATLAB'ın iki tür döngüsü vardır. For-end döngülerde geçiş sayısı döngü başladığında belirlenir. While-end döngülerde geçiş sayısı önceden bilinmez ve döngü süreci, belirli bir koşul karşılanana kadar devam eder. Her iki tür döngü de herhangi bir zamanda break komutu ile sonlandırılabilir.

for-end Döngüleri:

for-end döngülerde, bir komutun veya bir grup komutun yürütülmesi önceden belirlenen sayıda tekrarlanır. Bir döngünün şekli Şekil 'de gösterilmiştir.

• Döngü indeksi değişkeninin herhangi bir değişken adı olabilir (genellikle i, j, k, m ve n kullanılır, ancak MATLAB karmaşık sayılarla kullanılıyorsa i ve j kullanılmamalıdır).

```
for k = f:s:t

.....

.....Bir grup MATLAB kodu

.....

end
```

burada k indeksinin değeri f den t ye s artış miktarı kadar değişmektedir.

- İlk geçişte k = f ve bilgisayar, for ve end komutları arasındaki komutları yürütür. Daha sonra program ikinci geçiş için for komutuna geri döner. k, k = f + s'ye eşit yeni bir değer elde eder ve for ve end komutları arasındaki komutlar yeni k değeri ile yürütülür. Süreç son geçişe kadar kendini tekrar eder, burada k = t. Daha sonra program for'a geri dönmez, ancak son komutunu takip eden komutlarla devam eder. Örneğin, k = 1: 2: 9 ise, beş geçiş vardır ve karşılık gelen k değerleri 1, 3, 5, 7 ve 9'dur.
- Artış s negatif olabilir (yani; k = 25: –5: 10, k = 25, 20, 15, 10 ile dört geçiş üretir).
- Artış değeri s atlanırsa, değer 1'dir (varsayılan) (yani; k = 3: 7, k = 3, 4, 5, 6, 7 ile beş geçiş üretir).
- f = t ise, döngü bir kez yürütülür.
- Eğer f> t ve s> 0 ise veya f <t ve s <0 ise, döngü yürütülmez.
- k, s ve t'nin değerleri, k'nin t'ye eşit olamayacağı şekildeyse, o zaman s pozitifse, son geçiş, k'nin t'den küçük olan en büyük değere sahip olduğu yerdir (yani, k = 8: 10:50, k = 8, 18, 28, 38, 48 ile beş geçiş üretir). Eğer s negatifse, son geçiş k'nin t'den büyük en küçük değere sahip olduğu geçiştir.
- For komutunda k'ye belirli bir değer de atanabilir (vektör olarak yazılır). Örnek: k = [7 9 –1 3 3 5] için.
- k değeri döngü içinde yeniden tanımlanmamalıdır.
- Bir programdaki her komut için bir son komutu olmalıdır.
- Döngü indeksi değişkeninin (k) değeri otomatik olarak görüntülenmez. Döngüdeki komutlardan biri olarak k yazarak her geçişteki (bazen hata ayıklama için yararlı olan) değeri görüntülemek mümkündür.
- Döngü sona erdiğinde, döngü indeksi değişkeni (k) kendisine en son atanan değere sahiptir.
- Örnek

```
for k=1:3:10
x = k^2
end
```

Bu program yürütüldüğünde, döngü dört kez yürütülür. Dört geçişteki k değeri k = 1, 4, 7 ve 10'dur, bu da geçişlerde x'e atanan değerlerin sırasıyla x = 1, 16, 49 ve 100 olduğu anlamına gelir. İkinci satırın sonuna noktalı virgül yazılmadığından, x'in değeri her geçişte Komut Penceresinde görüntülenir. Komut dosyası yürütüldüğünde, Komut Penceresindeki görüntü

```
>> x =
1
x =
16
x =
49
x =
100
```

Örnek Seri Toplamı

$\sum_{k=1}^{n} {(-1)^k k}/{2^k}$ Dizisinin ilk n terimlerinin toplamını hesaplamak için komut dosyasında for-end döngüsü kullanın: n = 4 ve n = 20 için komut dosyasını çalıştırın.

Aşağıda serinin ilk n terimlerinin toplamını hesaplayan bir komut dosyası gösterilmektedir.

```
n=input('terim sayisini giriniz' );
S=0;   %toplamı sıfıra atıyoruz
for k=1:n
S=S+(-1)^k*k/2^k;  %Her geçişte serinin bir öğesi hesaplanır ve önceki geçişlerdeki öğelerin
%toplamına eklenir.
end
fprintf('serinin toplami: %f',S)
```

for-end döngüler ve öğe öğe işlemler hakkında bir not:

Bazı durumlarda, aynı sonuç, for-end döngüleri kullanılarak veya eleman-eleman işlemleri kullanılarak elde edilebilir. Örnek Problem, döngünün nasıl çalıştığını gösterir, ancak sorun, öğe öğe işlemler kullanılarak da çözülebilir. Dizilerle öğe öğe işlemler, MATLAB'ın aksi takdirde döngü gerektiren durumlarda hesaplama için araçlar sağlayan üstün özelliklerinden biridir. Genel olarak, öğe-öğe işlemler döngülerden daha hızlıdır ve her iki yöntem de kullanılabildiğinde önerilir.

Örnek Problem

Bir vektör V = [5, 17, –3, 8, 0, –7, 12, 15, 20, –6, 6, 4, –7, 16] ile verilir. Bir programı, pozitif olan ve 3 veya 5'e bölünebilen öğeleri ikiye katlayan ve negatif ancak –5'ten büyük olan öğeleri 3'ün üssüne yükselten bir komut dosyası olarak yazın.

Çözüm:

Sorun, if-elseif-end'e sahip bir for-end döngüsü kullanılarak çözülür.

içinde koşullu ifade. Geçişlerin sayısı, vektördeki öğelerin sayısına eşittir. Her geçişte bir öğe koşullu ifadeyle kontrol edilir. Sorun ifadesindeki koşulları yerine getirirse öğe değiştirilir. Gerekli işlemleri gerçekleştiren komut dosyası dosyasındaki bir program:

```
V=[5, 17, -3, 8, 0, -7, 12, 15, 20 -6, 6, 4, -2, 16];
n=length(V);
for k=1:n
if V(k)>0 & (rem(V(k),3) = = 0 | rem(V(k),5) = = 0)
V(k)=2*V(k);
elseif V(k) < 0 & V(k) > -5
V(k)=V(k)^3;
end
end
V
```

Çıktısı

```
V =
10 17 -27 8 0 -7 24 30 40 -6 12 4
-8 16
```

while-end döngüleri:

while-end döngüleri, döngü gerektiğinde kullanılır, ancak geçiş sayısı önceden bilinmemektedir. while-end döngülerinde döngü işlemi başladığında geçiş sayısı belirtilmez. Bunun yerine, döngü süreci, belirtilen koşul yerine getirildiği sürece devam eder. Bir while-end döngüsünün yapısı şu şekkildedir.

While koşullu ifade

……..

……. MATLAB komut grubu

…….

end

İlk satır, koşullu bir ifade içeren bir while ifadesidir. Program bu satıra ulaştığında koşullu ifade kontrol edilir. Yanlış (0) ise, MATLAB son ifadeye atlar ve programla devam eder. Koşullu ifade doğru (1) ise, MATLAB while ve end komutları arasında izleyen komut grubunu yürütür. Ardından MATLAB, while komutuna geri döner ve koşullu ifadeyi kontrol eder. Bu döngü süreci, koşullu ifade yanlış olana kadar devam eder.

While-end döngüsünün düzgün şekilde yürütülmesi için:

- while komutundaki koşullu ifade en az bir değişken içermelidir.
- MATLAB, while komutunu ilk kez çalıştırdığında koşullu ifadedeki değişkenler atanmış değerlere sahip olmalıdır.
- Koşullu ifadedeki değişkenlerden en az birine while ile bitiş arasındaki komutlarda yeni bir değer atanmalıdır. Aksi takdirde, döngü başladığında, koşullu ifade doğru kalacağı için asla durmayacaktır.

Aşağıdaki programda basit bir while-end döngüsü örneği gösterilmektedir. Bu programda, başlangıç değeri 1 olan bir x değişkeni, değeri 15'e eşit veya daha küçük olduğu sürece her döngüde ikiye katlanır.

```
x=1
while x<=15
x=2*x
end
```

Çıktısı

```
x =
1
x =
2
x =
4
x =
8
x =
16
```

x = 16 olduğunda, while komutundaki koşullu ifade yanlıştır ve döngü durur.

Önemli Not:

Bir while-end döngüsü yazarken, programcı, koşullu ifadede bulunan ve döngü işlemi sırasında yeni değerler atanan değişkenlerin (veya değişkenlerin) en sonunda while komutundaki yanlış koşullu ifadeyi oluşturan değerler atanacağından emin olmalıdır. Aksi takdirde döngü sonsuza kadar devam eder (belirsiz döngü). Yukarıdaki örnekte, koşullu ifade x> = 0.5 olarak değiştirilirse, döngü sonsuza kadar devam edecektir. Geçişlerin sayısı büyük bir değeri aşarsa geçişleri sayarak ve döngüyü durdurarak böyle bir durumdan kaçınılabilir. Bu, koşullu ifadeye maksimum geçiş sayısını ekleyerek veya kesme komutunu kullanarak yapılabilir.

Hiç kimse hata yapmaktan özgür olmadığından, dikkatli programlamaya rağmen belirsiz bir döngü durumu ortaya çıkabilir. Böyle bir durumda kullanıcı Ctrl + C veya Ctrl + Break tuşlarına basarak belirsiz bir döngünün yürütülmesini durdurabilir.

Örnek Problem: Bir fonksiyonun Taylor serisi gösterimi

$f(x) = e^x$ fonksiyonu Taylor serisinde $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$ ile temsil edilebilir.

Taylor serisi gösterimini kullanarak e^x i belirleyen bir MATLAB dosyasına bir program yazın. Program, serinin terimlerini toplayarak ve en son eklenen terimin mutlak değeri 0.0001'den küçük olduğunda durarak hesaplar. Bir while-end döngüsü kullanın, ancak geçiş sayısını 30 ile sınırlayın. 30. geçişte eklenen terimin değeri 0.0001'den küçük değilse, program durur ve 30'dan fazla terimin gerekli olduğunu belirten bir mesaj görüntüler.

Programı kullanarak $e^2, e^{-4}\ ve\ e^{21}$ i hesaplayınız.

Çözüm:

Taylor serisinin ilk birkaç terimi

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Fonksiyonu hesaplamak için seriyi kullanan bir program aşağıda gösterilmektedir. Program, kullanıcıdan x değerini girmesini ister. Daha sonra birinci terim olan an'a 1 sayısı atanır ve an toplamı S'ye atanır.Daha sonra, ikinci terimden itibaren program, serinin n'inci terimini hesaplamak ve toplama eklemek için while döngüsü kullanır. . Program ayrıca n terimlerin sayısını da sayar. While komutundaki koşullu ifade, n'inci terimin mutlak değeri 0.0001'den büyük olduğu ve geçişlerin sayısı 30'dan küçük olduğu sürece doğrudur. Bu, 30. terim 0.0001'den küçük değilse, döngü durur.

```
x=input('Enter x ' );
n=1; an=1; S=an;
while abs(an) >= 0.0001 & n <= 30
an=x^n/factorial(n);
S=S+an;
n=n+1;
end
if n >= 30
disp('More than 30 terms are needed')
else
fprintf('exp(%f) = %f',x,S)
fprintf('\nThe number of terms used is: %i',n)
```

end

Program, sonuçları görüntülemek için bir if-else-end ifadesi kullanır. Döngü 30. terim 0.0001'den küçük olmadığı için durduysa, bunu belirten bir mesaj görüntüler. İşlevin değeri başarıyla hesaplanırsa, işlevin değerini ve kullanılan terimlerin sayısını görüntüler. Program çalıştığında, geçiş sayısı x'in değerine bağlıdır. Program (expox olarak kaydedilen) e^2, e^{-4} *ve* e^{21} hesaplamak için kullanılır

```
>> expox
Enter x 2

exp(2.000000) = 7.389046

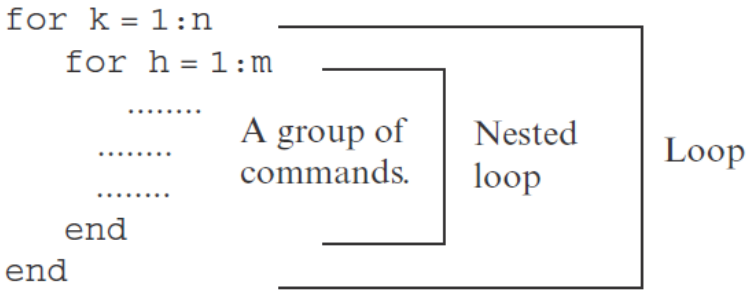
The number of terms used is: 12

>> expox
Enter x -4
exp(-4.000000) = 0.018307
The number of terms used is: 18

>> expox
Enter x 21
More than 30 terms are needed
```

İç İçe Döngüler Ve İç İçe Şartlı Cümleler

Döngüler ve koşullu ifadeler, diğer döngülerin veya koşullu ifadelerin içine yerleştirilebilir. Bu, bir döngünün ve / veya koşullu ifadenin başka bir döngü veya koşullu ifade içinde başlayabileceği (ve bitebileceği) anlamına gelir. Yuvalanabilecek döngülerin ve koşullu ifadelerin sayısında herhangi bir sınır yoktur. Bununla birlikte, her if, case, for ve while ifadesinin karşılık gelen bir bitiş ifadesine sahip olması gerektiği unutulmamalıdır. Şekil 6-7, başka bir for-end döngü içindeki yuvalanmış bir for-end döngünün yapısını gösterir. Bu şekilde gösterilen döngülerde, örneğin, n = 3 ve m = 4 ise, o zaman önce k = 1 ve iç içe döngü h = 1, 2, 3, 4 ile dört kez çalıştırılır.



Daha sonra k = 2 ve yuvalanmış döngü h = 1, 2, 3, 4 ile dört kez çalıştırılır. Son olarak k = 3 ve yuvalanmış döngü tekrar dört kez çalıştırılır. İç içe bir döngü her yazıldığında, MATLAB yeni döngüyü dış döngüye göre otomatik olarak girintiler. İç içe geçmiş döngüler ve koşullu ifadeler aşağıdaki örnek problemde gösterilmektedir.

break *ve* **continue** *Komutları*

Break komutu:

- Bir döngünün içindeyken (for veya while), break komutu döngünün yürütülmesini sonlandırır (yalnızca son geçişi değil, tüm döngü). Break komutu bir döngüde göründüğünde, MATLAB döngünün bitiş komutuna atlar ve bir sonraki komutla devam eder (o döngünün for komutuna geri dönmez).
- Break komutu iç içe bir döngünün içindeyse, yalnızca iç içe döngü sonlandırılır.
- Bir komut dosyası veya işlev dosyasındaki bir döngünün dışında bir break komutu göründüğünde, dosyanın yürütülmesini sonlandırır.
- Break komutu genellikle bir koşullu ifade içinde kullanılır. Döngülerde, örneğin bir koşul karşılanırsa, döngü sayısı önceden belirlenmiş bir değeri aşarsa veya bazı sayısal prosedürdeki bir hata önceden belirlenmiş bir değerden küçükse döngü sürecini sonlandırmak için bir yöntem sağlar. Bir döngünün dışında yazıldığında, break komutu, bir işlev dosyasına aktarılan verilerin beklenenle tutarlı olmaması gibi, bir dosyanın yürütülmesini sonlandırmak için bir yol sağlar.

continue komutu:

- continue komutu, mevcut geçişi durdurmak ve döngü sürecinde bir sonraki geçişi başlatmak için bir döngü içinde (for veya while) kullanılabilir.
- continue komutu genellikle bir koşullu ifadenin bir parçasıdır. MATLAB, devam komutuna ulaştığında, döngüdeki kalan komutları yürütmez, ancak döngünün bitiş komutuna atlar ve ardından yeni bir geçiş başlatır.

ÖDEV SETİ 6

1)
```
a=input(‘ilk sayiyi giriniz:’);
b= input(‘ikinci sayiyi giriniz:’);
c= input(‘yapmak istediginiz islemi seciniz: ”+, -, *, /”,’s’);
If c==’+’
Sonuc=a+b;
elseif c==’-’
Sonuc=a-b;
elseif c==’*’
Sonuc=a*b;
elseif c==’/’
  if  b~=0
Sonuc=a/b;
else
disp(‘b degeri sifir olamaz’)
end
else
disp(‘gecersiz sembol girdiniz’)
end
```

2) Yıkarıdaki işlemi swich case yapısını kullanarak yazınız.
   Örneğin;
   swich(c)
   case(‘+’)
   Sonuc=a+b;
   case(‘-’)
   Sonuc=a-b;
   .
   .
   .
    if b~=0
   Sonuc=a/b;
   else
   disp(‘bolen sayi sifir olamaz)
   end

   Yapısını
3) %For döngülü
   b=0;
   for a=1:10
   b=b+1
   end
   b ye 10 defa 1 eklenmiş olur
   aynı işlemi b ye 10 defa 2 ekleyerek yapınız.

4)
   b=0;
   c=0;
   for a=1:10

…

end

   …. Yerine b=b+1; c=b+a;
   daha sonara
   c=c+a^2
   yazarak işlemleri nceleyiniz.

5) Ekrana 100 defa adınızı yazdırınız.
   Örnek
   for a=1:100
   disp(‘adınız ‘)
   end
6) Bir sayının faktöriyelini alan bir program yazınız öneğin;
   a=input(‘bir sayı giriniz’)

```
sonuc=1;
for b=1:a
sonuc=sonuc*b
end
```

7) 6 da verilen faktöriyel alma programını girilen sayı sıfırsa sonucun 1 olduğu, negatifse faktöriyeli alınamaz diyen bir programa dönüştürünüz.
8) A 10 dan küçük olduğu müddetçe B yi 1 arttıran bir program whie döngüsü bir şarta bağlıdır ve o şart kod içinde yazılır. Örneğin;

```
A=0;
B=0;
while A<10
B=B+1;
A=A+1;
end
```