# FIZ219_EST_UygulamaNotlari_02_BasitDegiskenler

October 18, 2020

## 1 Uygulama Notları:2

### 1.1 FİZ219 - Bilgisayar Programlama I | 16/10/2020

- Değişken Türleri
- Değişken Tanımlama
- İşlem Türleri
- İşlem Öncelikleri
- Algoritmalar & Sanki-kod ('pseudo-code')

Emre S. Tasci emre.tasci@hacettepe.edu.tr

### 1.2 Değişken Türleri

GNU Octave'da değişkenler belli başlı iki ana sınıfa ayrılabilir: * Sayısal Değişkenler * String ('sözcük') Değişkenleri

Mühendislik hesaplarında yoğun olarak sayılarla işlem yapacağımızdan, dönem sonuna kadar string değişkenlerini bir kenara koyuyoruz (yine de çıktılarımızı güzelleştirmek için, çok gerekli olmasa da, aralarda kaçamak yapacağız 8).

Sayısal değişkenler, pozitif, negatif, kompleks, rasyonel, çok büyük, çok küçük değerler olabilir.

Değişkenlerin alabileceğin minimum ve maksimum değerler hakkında şu şekilde bilgi alabiliriz:

[1]:
```
realmin, realmax
```

```
ans =   2.2251e-308
ans =   1.7977e+308
```

[2]:
```
intmin, intmax
```

```
ans = -2147483648
ans = 2147483647
```

[3]:
```
eps
```

```
ans =    2.2204e-16
```

Daha emeklemeden koşmaya başladık, yukarıdakiler karışık geldiyse, sıkıntı yok, sâfi teknik bilgidir rahatlıkla geçebilirsiniz. Herhangi bir komut hakkında bilgi almak içinse `help` yazıp, komutu girin, örneğin:

[4]:
```
help eps
```

```
'eps' is a built-in function from the file libinterp/corefcn/data.cc

 -- eps
 -- eps (X)
 -- eps (N, M)
 -- eps (N, M, K, …)
 -- eps (…, CLASS)
     Return a scalar, matrix or N-dimensional array whose elements are
     all eps, the machine precision.

     More precisely, 'eps' is the relative spacing between any two
     adjacent numbers in the machine's floating point system.  This
     number is obviously system dependent.  On machines that support
     IEEE floating point arithmetic, 'eps' is approximately 2.2204e-16
     for double precision and 1.1921e-07 for single precision.

     When called with no arguments, return a scalar with the value 'eps
     (1.0)'.

     Given a single argument X, return the distance between X and the
     next largest value.

     When called with more than one argument the first two arguments are
     taken as the number of rows and columns and any further arguments
     specify additional matrix dimensions.  The optional argument CLASS
     specifies the return type and may be either "double" or "single".

     See also: realmax, realmin, intmax, flintmax.

Additional help for built-in functions and operators is
available in the online version of the manual.  Use the command
'doc <topic>' to search the manual index.

Help and information about Octave is also available on the WWW
at https://www.octave.org and via the help@octave.org
mailing list.
```

… demek ki `eps` komutu bize makinemizin hassasiyetini veriyormuş.

### 1.3 Değişken Tanımla (Atama)

Değişkenleri tanımlamak, ya da programlama diliyle söylersek, onlara "değer atamak" için `=` operatörünü kullanırız:

[5]:
```
a = 5
```

```
a =  5
```

[6]:
```
b = 3.14
```

```
b =  3.1400
```

bir de şuna bakalım:

[7]:
```
c = 5.0
```

```
c =  5
```

Yukarıda `a`'yı tam sayı olarak tanımladık, `b`'yi ise ondalıklı sayı, Octave da o şekilde kabul etti. Ama `c`'yi tam sayı olarak değil de, ondalıklı sayı olarak tanımlamaya kalkınca, hiç oralı olmadı. Dahası:

[8]:
```
3.4 - 1.4
```

```
ans =  2
```

şeklinde iki ondalıklı sayıyı çıkarınca da, sonucu ondalıklı cinsten değil de, tam sayı cinsinden verdi. Bir sayının tam sayı mı, yoksa ondalıklı sayı mı olduğunun kararını neye göre veriyor? (cevap: aslında biz özellikle belirtmediğimiz sürece, hepsini ondalıklı olarak tutuyor, nasıl belirteceğimizi de ileride göreceğiz ama şimdilik hiç sorun yok 8)

Peki ya bu nasıl bir şey?

[9]:
```
d = pi
```

```
d =  3.1416
```

`pi`yi halihazırda biliyormuş! Peki hassasiyet az değil mi? Değil. Aslında sayıyı kendinde tuttuğu hassasiyet, bize gösterdiği hassasiyetle aynı değil (çok karmaşık şeylerle ekran dolmasın diye bize azını gösteriyor). Eğer daha hassas görmek isterseniz `format` komutu size göre:

[10]:
```
format long
```

[11]:
```
d
```

```
d =  3.141592653589793
```

"normal" duruma `format short` ile dönebiliriz.

[12]:
```
format short
d
```

```
d =  3.1416
```

Daha da hassas gösterimi mümkün ama bize şimdilik bu kadarı da yeter. Aklınızda tutmanız gereken, sayının değerinin ille de size sunulduğu hassasiyette olmadığı...

Hazır `pi`yi görmüşken, birkaç tane daha özel matematik sabit tanımlı olarak geliyor, bunlar: Euler'in sayısı (`e`), sonsuz (`inf`), -1'in kökü (`I`), ama daha onlara da var (sabır!.. ;)

`a`'yı tanımladık, peki `x`'i `a` cinsinden tanımlayıp, sonra `a`'nın değerini tekrar tanımlayalım: bakalım `x`'in değeri değişecek mi?

[13]:
```
a = 5
```

```
a =  5
```

[14]:
```
x = a
```

```
x =  5
```

[15]:
```
a = 3
```

```
a =  3
```

[16]:
```
x
```

```
x =  5
```

(bir değişkenin sadece adını yazıp çalıştırdığımızda, o değişkenin değeri yazılır).

### 1.4 İşlem Türleri

Octave'da pek çok çeşit işlem türü vardır, bunlardan atamayı, çıkartmayı ve hatta fonksiyonu gördük bile (hangi fonksiyonu gördük acaba?). Aritmetik işlemleri çoklukla kullanacağız, fonksiyonları da önümüzdeki haftadan itibaren yavaş yavaş kullanmaya başlayacağız.

[17]:
```
3 + 4
```

```
ans =  7
```

[18]:
```
a
```

```
a =  3
```

[19]:
```
a + 4
```

```
ans =  7
```

[20]:
```
b = 4
```

```
b =  4
```

[21]:
```
a + b
```

```
ans =  7
```

Görüldüğü üzere, değişken olsun, değer olsun, iki çeşit birbirleriyle güzelce harmanlanabiliyor, işleme girebiliyor. Çıkarma ve toplamadan başka çokça kullanacağımız diğer başlıca işlemler: * Çarpma operatörü: `*` * Bölme operatörü: `/` * Üs alma operatörü: `**` (`^` de kullanılır)

[22]:
```
3 * 5
```

```
ans =  15
```

[23]:
```
3 / 5
```

```
ans =  0.60000
```

[24]:
```
5**3
```

```
ans =  125
```

[25]:
```
5^3
```

```
ans =  125
```

Skaler işlemlerde nasıl çıkarmayı toplama, bölmeyi çarpma cinsinden yazabiliyorsak, kök almayı da üs alma cinsinden yazabiliriz. Demek istediğim:

[26]:
```
5 + (-3)
```

```
ans =  2
```

[27]:
```
3 * (1/5)
```

```
ans =  0.60000
```

[28]:
```
125 ^ (1/3)
```

```
ans =  5.0000
```

İşlemleri hızlandırmak adına, üs ve özelde karekök alma işlemlerine ayrı komutlar atanmıştır:

[29]:
```
power(5,3)
```

```
ans =  125
```

[30]:
```
sqrt(25)
```

```
ans =  5
```

[31]:
```
nthroot(125,3)
```

```
ans =  5
```

Bu türden "yeniden/fazladan" tanımlamaların sebebi, sıklıkla kullanıldığı için, "optimize" edilmiş olmalarıdır (bu, 5'in karesini hesaplamak için `power(5,2)` yazmak yerine, hatta `5^2` yazmak yerine kısaca `5*5` yazmaya benzer 8)

## 1.5 İşlem Öncelikleri

**Örnek:**

$$a + \frac{b+c}{(d+e)^2}$$

işlemini elde ve Octave'da yapın – birbirini tutuyor mu? ;)

[32]:
```
a = 5; b = 3; c = -4; d = 6; e = 1;
```

(Bir satırda birden fazla işlem yaptırmak istediğimizde, işlemleri ; ile ayırabildiğimiz gibi, bunu aynı zamanda çıktıyı "bastırmak" (*suppress*) için de kullanıyoruz).

[33]:
```
a + (b+c) / (d+e)^2
```

```
ans =  4.9796
```

Ama yine de işleri garantiye ve (grupları paranteze) almak her zaman iyidir:

[34]:
```
a + ((b+c) / (d+e)^2)
```

```
ans =  4.9796
```

### 1.5.1 Resmi işlem öncelik sırası

1. '()' '{}' '.'
2. '++' '–'
3. '" '.' '^' ' ' '.^' '.'
4. '+' '-' '++' '–' '~' '!'
5. ' ' '/' ' ' '.' '.' './'
6. '+' '-'

...

'='

Kaynak.)

## 1.6 Algoritmalar & Sanki-kod ('pseudo-code')

### 1.6.1 Ortalama sanki-kod'u

1. a'nın değerini tanımla.
2. b'nin değerini tanımla.
3. c'nin değerini a ve b'nin toplamı olarak ata.
4. c'nin değerini ikiye böl.

[35]:
```
# 1
a = 5

# 2
b = 3

# 3
c = a + b

# 4
c / 2
```

```
a =  5
b =  3
c =  8
ans =  4
```