

MA2151

Simulasi dan Komputasi Matematika

Dosen:

Novriana Sumarti, Ph.D.

Dr. Rinovia G. Simanjuntak

Prof. Dr. M. Wono Setya Budi



Bagian I: Pengenalan Pemrograman dengan Matlab

1. Penulisan simbolik
2. Fungsi handle
3. Animasi
4. Debugging



Penulisan simbolik

Penulisan variabel simbolik bertipe double atau char.

Apakah bedanya:

```
>> a = 4/5;
```

```
>> b = 1/3;
```

```
>> d = a*b
```

dengan

```
>> a = sym('4/5');
```

```
>> b = sym('1/3');
```

```
>> d = a*b
```

Bedanya

Yang pertama


$d = 0.2667 \rightarrow$ numerik

Yang kedua

$d = 4/15 \rightarrow$ simbolik

	Keuntungan	Kerugian
Simbolik	<ul style="list-style-type: none">- Solusi analitik- Dapat melihat bentuk fungsi solusinya	<ul style="list-style-type: none">- Kadang-kadang tidak ada solusi- Dapat berbentuk sangat kompleks
Numerik	<ul style="list-style-type: none">- Selalu mendapatkan solusi- Dapat membuat solusi akurat- Lebih mudah pemrogramannya	<ul style="list-style-type: none">- Sulit untuk menggali lebih dalam- Kadang-kadang memberi solusi salah- Dapat memakan waktu lama untuk pemrograman

```
>>c = sym('c','positive'); %mendefinisikan suatu variabel
>>expand((a-c)^2)
ans =
  c^2 - (2*c)/3 + 1/9
>>factor(ans)
ans =
(3*c - 1)^2/9
>>mat= sym([1 2 ; 3 4]);
>>matInv = inv(mat)
matInv =
 [ -2,  1]
 [ 3/2, -1/2]
>> collect(3*x+4*y-1/3*x^2-x+3/2*y)
```



```
>> syms x y real
>> collect(3*x+4*y-1/3*x.^2-x+3/2*y)
ans =
- x^2/3 + 2*x + (11*y)/2
>> simplify(cos(x)^2+sin(x)^2)
ans =
1
>> subs('c^2',c,5)
ans =
25
>> subs('c^2',c,x/2)
ans =
x^2/4
```



Penulisan simbolik matriks:

```
>> mat = sym('[a b ; c d]');
```

```
>> mat2 = mat*[1 3 ; 4 -2]
```

```
mat2 =
```

```
[ a + 4*b, 3*a - 2*b]
```

```
[ c + 4*d, 3*c - 2*d]
```

```
>> dt = det(mat)
```

```
dt =
```

```
a*d - b*c
```

```
>> iv = inv(mat)
```

```
iv =
```


```
[ d/(a*d - b*c), -b/(a*d - b*c)]
```

```
[ -c/(a*d - b*c), a/(a*d - b*c)]
```

```
>> iv(2,3)
```

```
ans =
```

```
-c/(a*d - b*c)
```



```
>> syms a b r x y
>> solve('(x-a)^2+(y-b)^2=r^2','x')
```

ans =

$$a + (b + r - y)^{(1/2)} * (r - b + y)^{(1/2)}$$
$$a - (b + r - y)^{(1/2)} * (r - b + y)^{(1/2)}$$

```
>> solve('(x-a)^2+(y-b)^2=r^2','y')
```

ans =

$$b + (a + r - x)^{(1/2)} * (r - a + x)^{(1/2)}$$
$$b - (a + r - x)^{(1/2)} * (r - a + x)^{(1/2)}$$



Fungsi handle

```
function[ y1, y2] = addval(m)
disp(m);
% memasukkan fungsi addone ke setiap elemen dari m
y1 = fungsiutama(@addone,m);
disp(y1);
% memasukkan fungsi addtwo ke setiap elemen dari m
y2 = fungsiutama(@addtwo,m);
disp(y2);
function y = addone(u)
y = u + 1;
end
function y = addtwo(u)
y = u + 2;
end
```



Function dapat dituliskan dalam 1 script dengan cara menuliskan **end**.

Contoh:

```
function y = pertama(x)
y = rand(2);
end
```

```
function y = kedua(x)
y = rand(3);
end
```



Animasi

MATLAB dapat merekam movie frames dan memutar ulang secara otomatis.

Format yang biasa digunakan:avi, animated gif

Menggambar frame demi frame

```
close all;  
for t = 1:30  
    imagesc(rand(200));  
    colormap(gray);  
    pause(.5);  
end
```



Suatu movie adalah deretan frame yang terekam:

```
clear all;  
close all;  
for t = 1:30  
    imagesc(rand(200));  
    colormap(gray);  
    M(t)=getframe;  
end
```

Untuk menampilkan kembali:

```
>> movie(M, 2, 30);  
loop movie 2 kali dengan 30 frame per detik
```



Debugging

Debugging adalah cara untuk mencek letak error atau kesalahan perhitungan akibat kesalahan penulisan program.

Saat mulai debugging, gunakan disp untuk mengeluarkan pesan sehingga kita tahu perhitungan sudah sampai dimana

```
>> disp('mulai loop')
```

```
>> disp('loop berakhir')
```

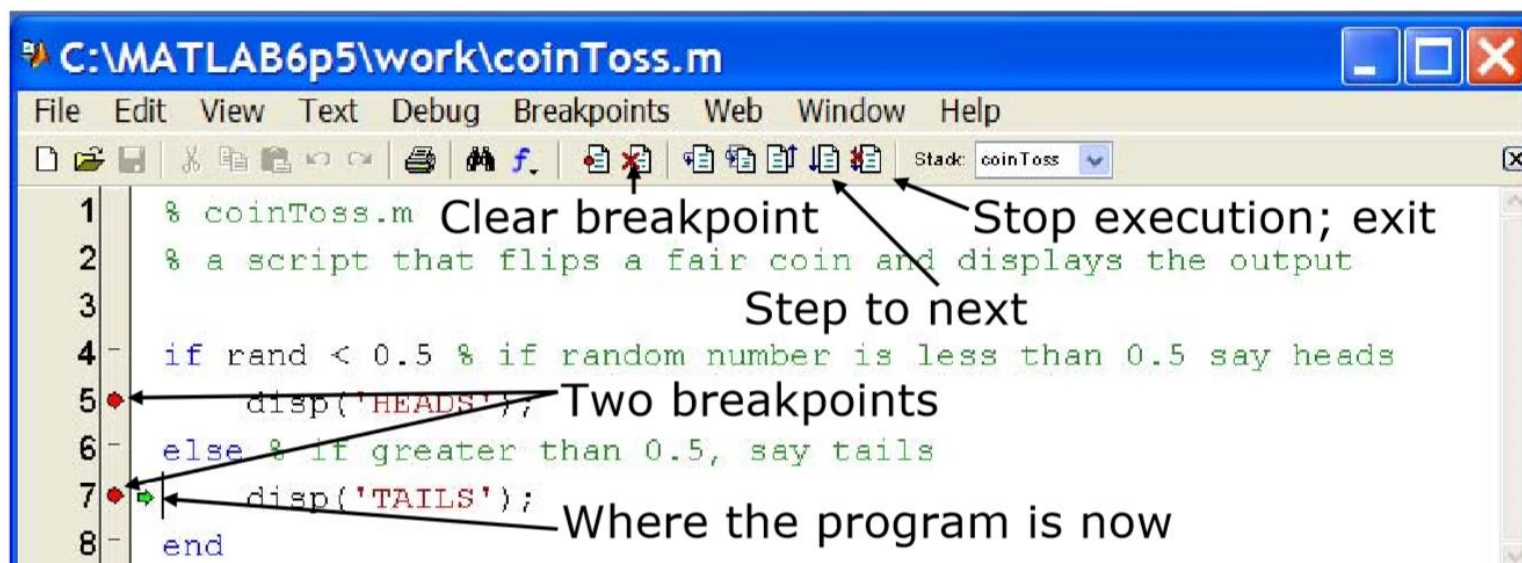
Kadang-kadang perlu untuk mengeluarkan nilai variabel

```
>> disp(strcat(['loop iteration ', num2str(n)]));
```

Strcat concatenates string yang diberikan

Debugging

- To use the debugger, set breakpoints
 - Click on – next to line numbers in MATLAB files
 - Each red dot that appears is a breakpoint
 - Run the program
 - The program pauses when it reaches a breakpoint
 - Use the command window to probe variables
 - Use the debugging buttons to control debugger



The screenshot shows the MATLAB editor window for a file named 'C:\MATLAB6p5\work\coinToss.m'. The window title bar includes standard Windows window controls (minimize, maximize, close) and the file path. The menu bar contains 'File', 'Edit', 'View', 'Text', 'Debug', 'Breakpoints', 'Web', 'Window', and 'Help'. Below the menu bar is a toolbar with various icons for file operations and debugging. The main area displays the script code with line numbers 1 through 8. Two red dots are placed to the left of lines 5 and 7, indicating breakpoints. Annotations with arrows point to these dots and to the 'Breakpoints' menu item. The annotations include: 'Clear breakpoint' pointing to the breakpoint on line 5, 'Stop execution; exit' pointing to the 'Breakpoints' menu, 'Step to next' pointing to the 'Breakpoints' menu, 'Two breakpoints' pointing to both red dots, and 'Where the program is now' pointing to the red dot on line 7. The script code is as follows:

```
1 % coinToss.m Clear breakpoint
2 % a script that flips a fair coin and displays the output
3
4 if rand < 0.5 % if random number is less than 0.5 say heads
5     disp('HEADS'); Two breakpoints
6 else % if greater than 0.5, say tails
7     disp('TAILS'); Where the program is now
8 end
```