Chapter 4: Loops

Yunong Zhang (张雨浓)

Email: zhynong@mail.sysu.edu.cn

Introduction to Loops

- Repeating a calculation for a number of times
- Two types of loops (In MATLAB)
  - ***while*** loop

    repeated an indefinite number of times until a user-specified condition is satisfied
  - ***for*** loop

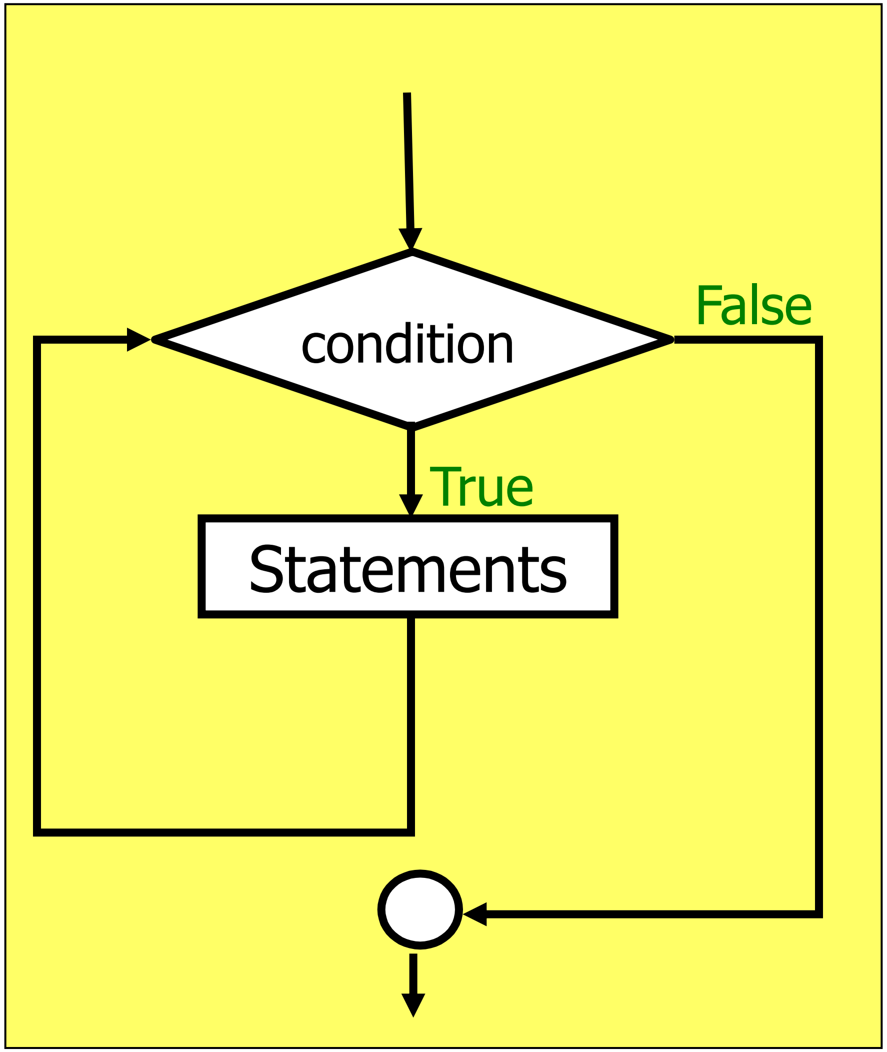
    repeated for a number of times, where the number of repetitions is user-specified and known

while loop

Looping process is terminated
based on the specified condition

while loop (Cont.)

```
while  condition
       statements
end
```



Without the 🟩 , there is no way to get out of the loop.

It's called "infinite loop"

while example (I)

cmd5.m

```
while k > 0
   disp(k);
end
```

Results

```
>> cmd5
??? Undefined function or variable "k".

Error in ==> cmd5 at 1
while k > 0
```

while example (II)

cmd6.m

```
k = 5;
while k > 0
    disp(k);
end
```

```
>> cmd6
   5
   5
   5
   5
  …
```

To break the program
Press “**Ctrl-C**”

while example (III)

cmd7.m

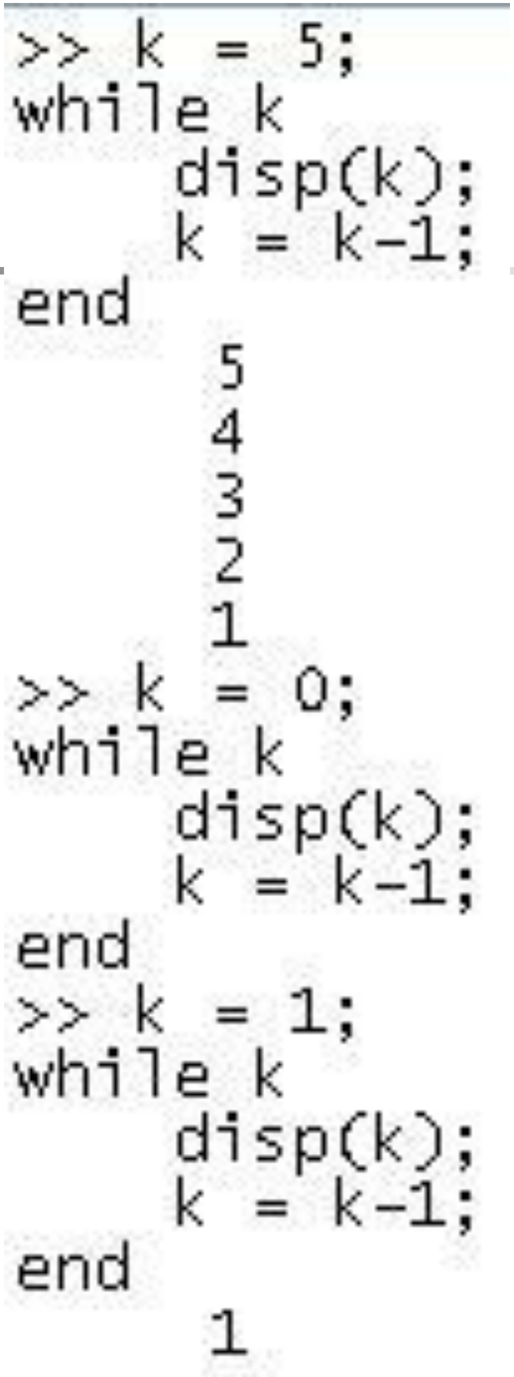
```
k = 5;
while k > 0
    disp(k);
    k = k-1;
end
```

Results

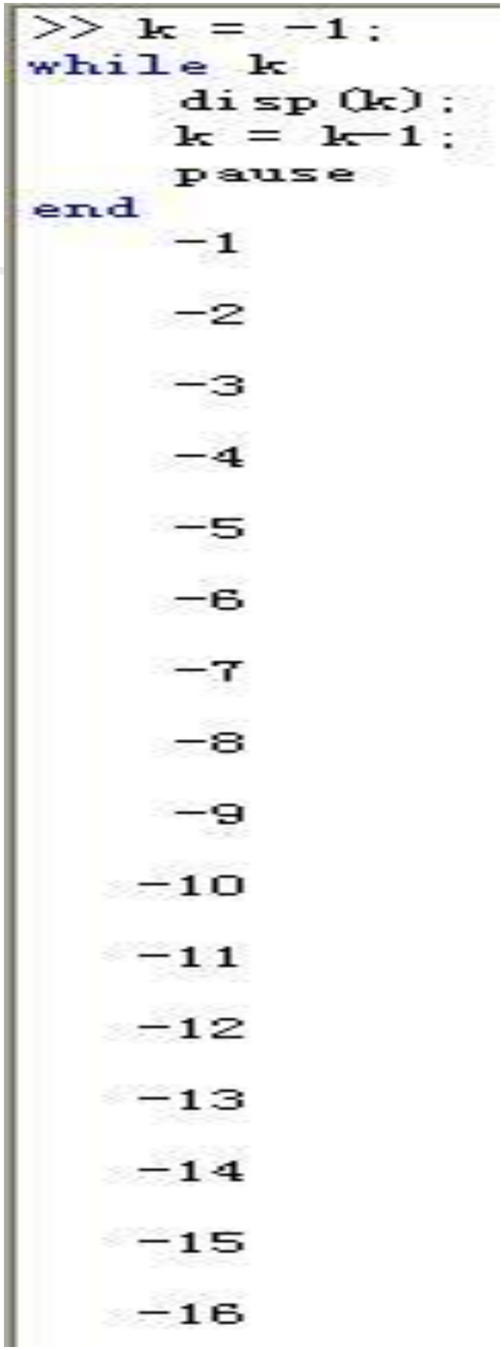
```
>> cmd7
     5
     4
     3
     2
     1
```

while exam:

```
k = 5; % 1, 0, 9, -1
while k
    disp(k);
    k = k-1;
end
```



```
>> k = 5;
while k
     disp(k);
     k = k-1;
end
     5
     4
     3
     2
     1
>> k = 0;
while k
     disp(k);
     k = k-1;
end
>> k = 1;
while k
     disp(k);
     k = k-1;
end
     1
```



```
>> k = -1;
while k
        disp(k);
        k = k-1;
        pause
end
        -1

        -2

        -3

        -4

        -5

        -6

        -7

        -8

        -9

       -10

       -11

       -12

       -13

       -14

       -15

       -16
```

while example (IV)

- Statistical Analysis
  - ***Arithmetic mean***

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

  - ***Standard deviation***

$$s = \sqrt{\frac{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}{N(N-1)}}$$

Median? 中位数，中值

Geometric mean?

Any problem? Alternative equations?

while example (IV) (Cont.)

- **State the problem**

Calculate the average and the standard deviation of a set of measurements;

All of the measurements >=0;

The number of the measurements in the data set is unknown.

while example (IV) (Cont.)

- **Define the inputs and outputs**

Inputs: Measurements whose values are >=0;

Outputs: Arithmetic mean
Standard deviation

while example (IV) (Cont.)

- **Design the algorithm**

  Input measurements;

  (Accumulate the input data);

  Calculate the number of measurements, the mean and standard deviation;

  Output the mean and standard deviation

while example (IV) (Cont.)

- **Turn the algorithm into Matlab statements**

```
% x                input-datum value
% n                number of input samples
% sum_x            sum of input values
% sum_x2           sum of the squares of samples
% xbar             average of samples
% std_dev          standard deviation of samples
```

while example (IV) (Cont.)

```
n=0;
sum_x=0;
sum_x2=0;

x=input (‘Enter the first value:’);
```

while example (IV) (Cont.)

```
while x>=0
     n=n+1;
    sum_x=sum_x+x;
    sum_x2=sum_x2+x^2;

    x=input (‘Enter the next value:’);
 end
```

while example (IV) (Cont.)

```
xbar=sum_x/n;
std_dev=sqrt((n*sum_x2-sum_x^2)/(n*(n-1)));

fprintf(‘The mean of the data set is: %f\n’, xbar)
fprintf(‘The standard deviation is: %f\n’, std_dev);
```

while example (IV) (Cont.)

- **Test the program**

```
>>cmd5.m
  Enter the first value: 3
  Enter the next value: 4
  Enter the next value: 5
  Enter the next value: -2
 The mean of the data set is: 4.000000
 The standard deviation is:1.000000
```

while example (IV) (Cont.)

- What will happen if we input

```
Enter the first value:3
Enter the next value:-1
```

while example (IV) (Cont.)

```
Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to
suppress this warning.)
> In C:\MATLAB6p5\work\cmd5.m at line 14
The mean of the data set is: 3.000000
The standard deviation is: NaN
```

while example (IV) (Cont.)

- **One way of modifying the program:**

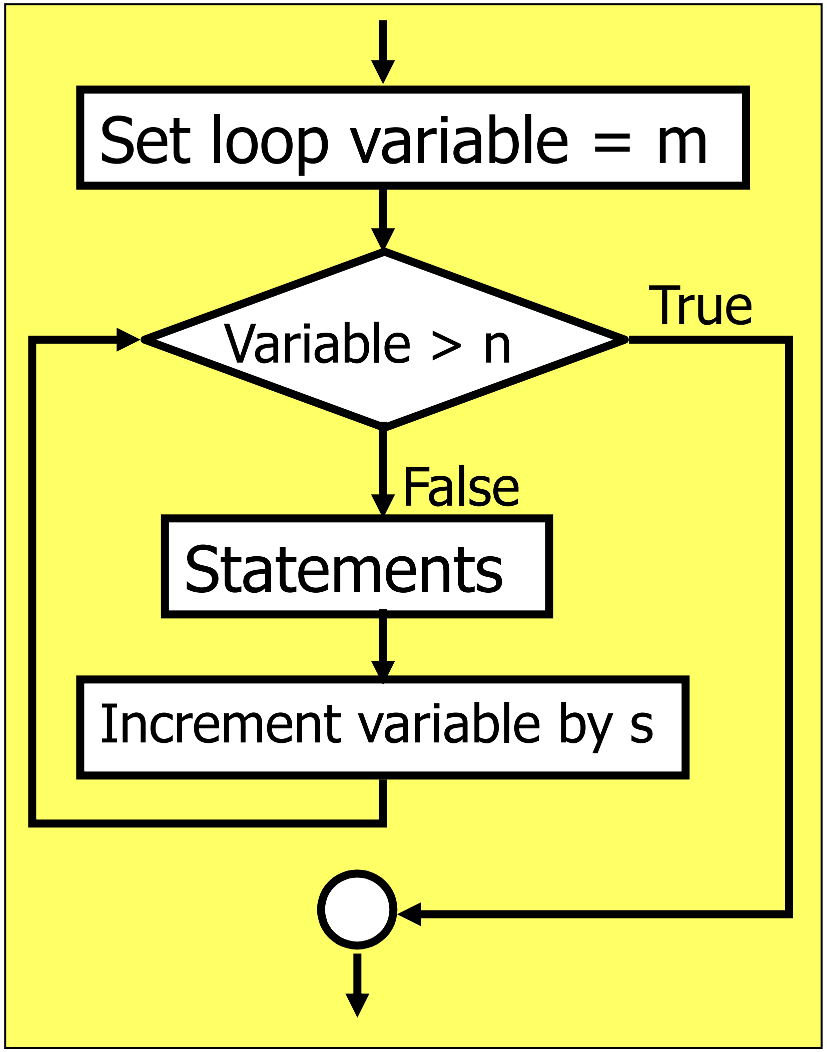
```
 while x>=0
        n=n+1;
        sum_x=sum_x+x;
        sum_x2=sum_x2+x^2;
        x=input (‘Enter the next value:’);
   end
if n==1
   disp(‘At least 2 values should be entered!’);
else
   xbar=sum_x/n;
   std_dev=sqrt((n*sum_x2-sum_x^2)/(n*(n-1)));

  fprintf(‘The mean of the data set is: %f\n’, xbar)
  fprintf(‘The standard deviation is: %f\n’, std_dev);
end
```

Other ways?

for loops

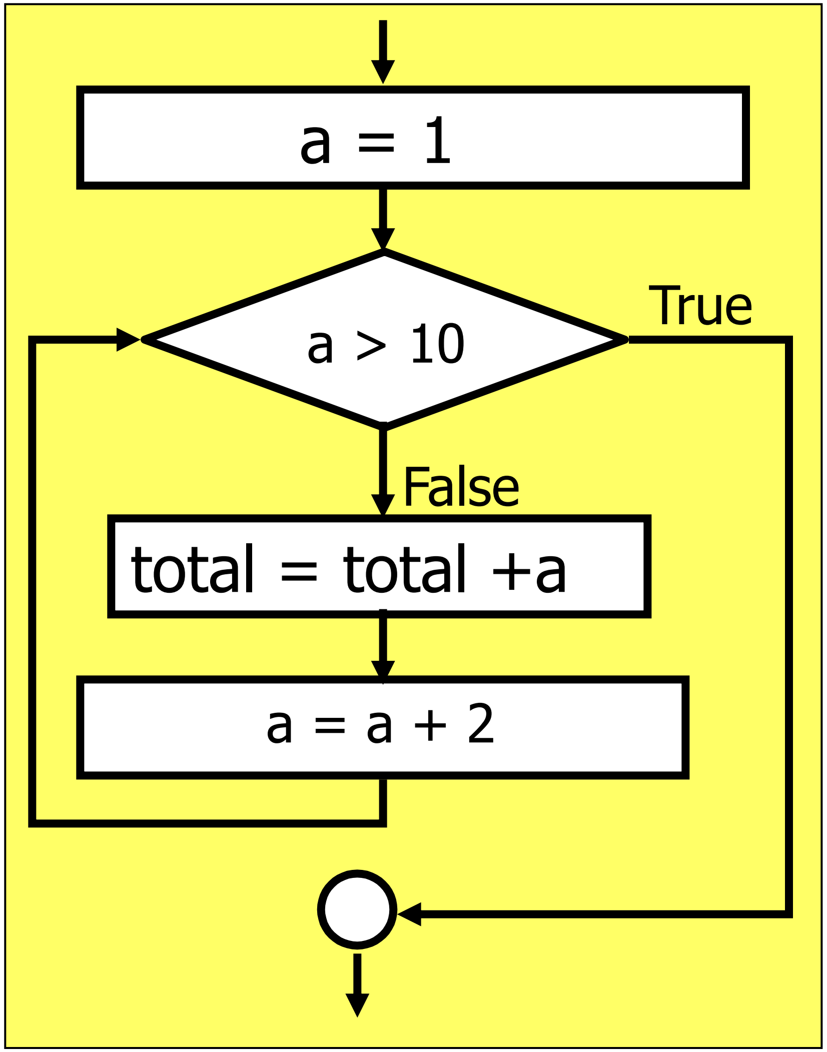
```
for variable = m:s:n
    statements
end
```



Note that the flowchart to the right is for the case of s>0.

for loops (Cont.)

```
for a = 1:2:10
    total = total + a
end
```



Examples: for loops

```
for variable = m:s:n
    statements
end
```

```
for a = 10:-2:0
    c = a * 2
end
```

```
for a = 1:10
    c = a * 2
end
```

```
for a = 10:2:5
    c = a * 2
end
```

```
for a = 2:2
    c = a * 2
end
```

```
for a = 2:1.5:10
    c = a * 2
end
```

```
for a = [5 9 7]
    c = a * 2
end
```

Examples: for loops (Cont.)

```
for a = 10:-2:0
      c = a * 2
end
```

```
a=10>=0,  c=20;
a=10-2=8>=0, c=16;
a=8-2=6>=0, c=12;
a=6-2=4>=0, c=8;
a=4-2=2>=0, c=4;
a=2-2>=0, c=0.
```

Examples: for loops (Cont.)

```
for a = 1:10
     c = a * 2
end
```

```
a=1<=10, c=2;
a=1+1=2<=10,c=4;
…
a=7+1=8<=10, c=16;
a=8+1=9<=10,c=18;
a=9+1=10<=10,c=20.
```

Examples: for loops (Cont.)

```
for a = 10:2:5
      c = a * 2
end
```

variable = m:s:n

if s>0, m should be <=n

10<=5? No!

The program is NOT executed!

Examples: for loops (Cont.)

```
for a = 2:2
      c = a * 2
end
```

a=2<=2, c=4.

Examples: for loops (Cont.)

```
for a = 2:1.5:10
     c = a * 2
end
```

```
a=2<=10, c=4;
a=2+1.5=3.5<=10, c=7;
a=3.5+1.5=5<=10, c=10;
a=5+1.5=6.5<=10,c=13;
a=6.5+1.5=8<=10,c=16;
a=8+1.5=9.5<=10,c=19.
```

Examples: for loops (Cont.)

```
for a = [5 9 7]
      c = a * 2
end
```

```
a=5, c=10;
a=9,c=18;
a=7,c=14.
```

Now return to Page 21 to give better flowcharts about the *for* loops.

Notes: for loops

```
for variable = m:s:n
    statements
end
```

- The loop variable should **NOT** be modified anywhere within the loop.

```
for a=1:10
    c=2*a;
    a=5;     %Infinite loop
end
```

Notes: for loops (Cont.)

CASE 1:

```
for i=1:100
    square(i)=i^2;
end
```

CASE 2:

```
square=zeros(1,100);
for i=1:100
    square(i)=i^2;
end
```

In CASE 1, the vector square has different size at different time. At each time, Matlab has to

1) create a new array/vector/matrix;
2) copy the contents of the old array to the new longer array;
3) add the new value to the array; and,
4) delete the old array.

CASE 2 is preferred!

Notes: for loops (Cont.)

Case A:

```
for i=1:100
    square(i)=i^2;
    square_root(i)=i^(1/2);
    cube_root(i)=i^(1/3);
end
```

100 *3 lines

Case B:

```
i=1:100;
square=i.^2;
square_root=i.^(1/2);
cube_root=i.^(1/3);
```

4 lines

Case B is preferred!

Comparing Loops and Vectorization

- Compare the execution speeds of loops and vectorized statements by performing and timing the following three sets of calculations

1. Calculate the square of every integer from 1 to 10,000 in a *for* loop without initializing the array of square first;
2. Calculate the square of every integer from 1 to 10,000 in a *for* loop, using the zeros function to pre-allocate the array of square first;
3. Calculate the square of every integer from 1 to 10,000 by vector operations.

Comparing Loops and Vectorization (Cont.)

- **Solution**

  tic: resets the built-in elapsed time counter

  toc: returns the elapsed time in seconds
  since the last call to function tic

滴答命令

Comparing Loops and Vectorization (Cont.)

```
% i             loop index
% square        array of squares
% average1      average time for calculation 1
% average2      average time for calculation 2
% average3      average time for calculation 3
```

Comparing Loops and Vectorization (Cont.)

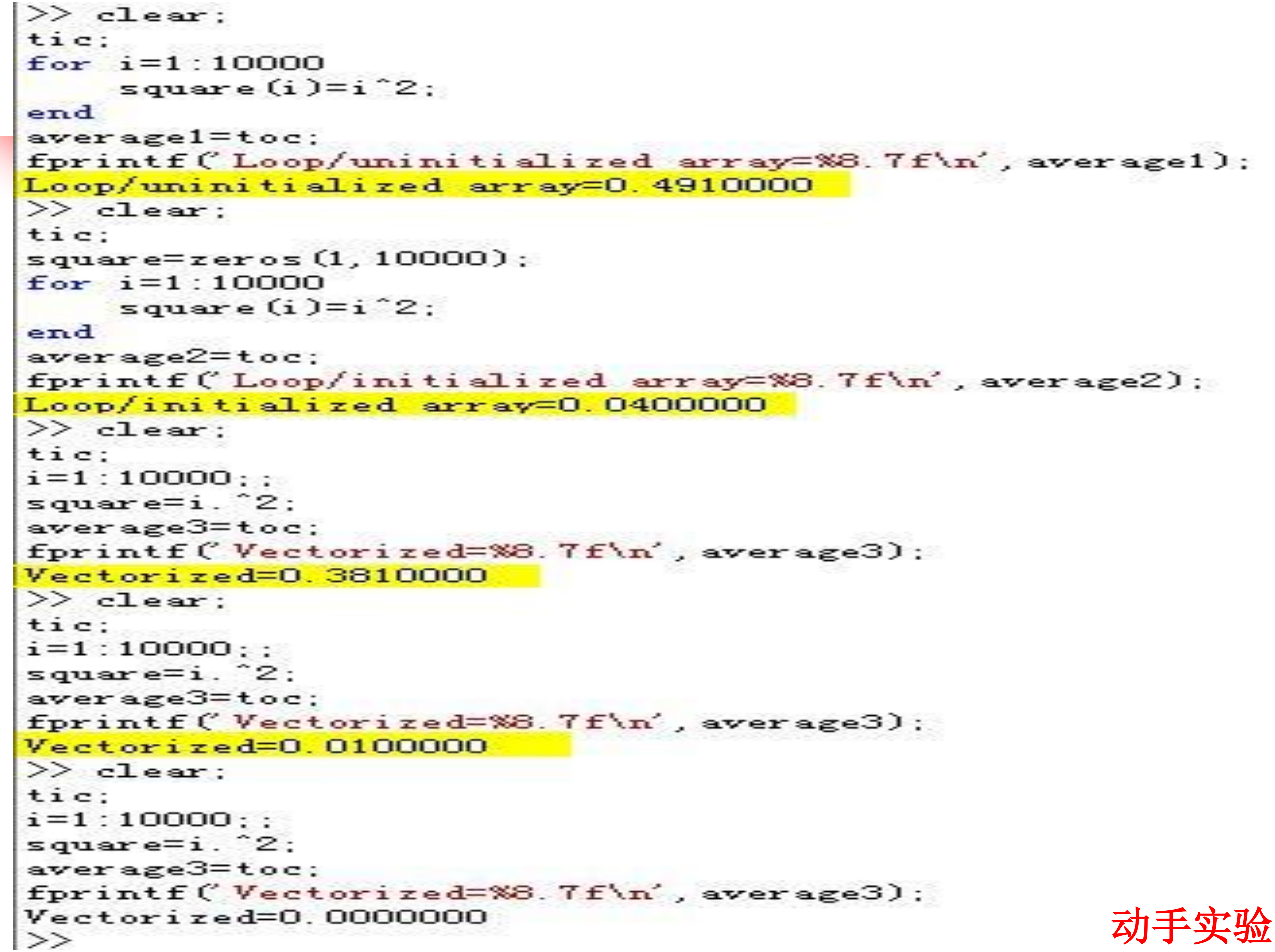
```
clear;
tic;
for i=1:10000
    square(i)=i^2;
end
average1=toc;
fprintf('Loop/uninitialized array=%8.7f\n',average1);
```

Comparing Loops and Vectorization (Cont.)

```
clear;
tic;
square=zeros(1,10000);
for i=1:10000
    square(i)=i^2;
end
average2=toc;
fprintf('Loop/initialized array=%8.7f\n',average2);
```

Comparing Loops and Vectorization (Cont.)

```
clear;
tic;
i=1:10000;;
square=i.^2;
average3=toc;
fprintf('Vectorized=%8.7f\n',average3);
```



```
>> clear;
tic;
for  i=1:10000
     square(i)=i^2;
end
average1=toc;
fprintf('Loop/uninitialized array=%8.7f\n',average1);
Loop/uninitialized array=0.4910000
>> clear;
tic;
square=zeros(1,10000);
for  i=1:10000
     square(i)=i^2;
end
average2=toc;
fprintf('Loop/initialized array=%8.7f\n',average2);
Loop/initialized array=0.0400000
>> clear;
tic;
i=1:10000;;
square=i.^2;
average3=toc;
fprintf('Vectorized=%8.7f\n',average3);
Vectorized=0.3810000
>> clear;
tic;
i=1:10000;;
square=i.^2;
average3=toc;
fprintf('Vectorized=%8.7f\n',average3);
Vectorized=0.0100000
>> clear;
tic;
i=1:10000;;
square=i.^2;
average3=toc;
fprintf('Vectorized=%8.7f\n',average3);
Vectorized=0.0000000
>>
```

动手实验

break command

To abnormally jump out of the loop before its end

If a *break* statement is executed in the body of a loop, the execution of the body will stop and control will be transferred to the first executable statement after the loop.

Find the answer of the program

cmd1.m

```
for num = 10:-2:0
   disp(num);
   temp = 2*num - 10;
   solution = temp + 5;
end
solution = solution - 10
```

Results

```
>> cmd1
    10
     8
     6
     4
     2
     0

solution =
   -15
```

break example

cmd2.m

```
% showing 'break' command
for num = 10:-2:0
    disp(num);
    temp = 2*num - 10;
    if (temp <= 0)
      break
    end
    solution = temp + 5;
end
solution = solution - 10
```

Results

```
>> cmd2
    10
     8
     6
     4

solution =
    -3
```

break == goto 09 ``solution=solution-10" in some programming languages.

break == jump-to 09 ``solution=solution-10" in some coding languages.

continue command

The *continue* statement jump from the current statement to the top of the loop.

continue example

cmd3.m

```
a = [100 0 10 -10];
for k = 1:length(a)
   solution = 1000/a(k)
end
```

Results

```
>> cmd3
solution =
    10
Warning: Divide
by zero.
> In cmd3 at 3
solution =
   Inf
solution =
   100
solution =
  -100
```

continue example (Cont.)

cmd4.m

```
% showing 'continue' command
a = [100 0 10 -10];
for k = 1:length(a)
   if (a(k)== 0)
     continue
   end
   solution = 1000/a(k)
end
```

Results

```
>> cmd4
solution =
    10
solution =
   100
solution =
  -100
```

break == goto 02 ``for k=1:length(a)” in some programming languages.

break == jump-to 02 `` for k=1:length(a)” in some coding languages.

Nesting Loops

Example:

```
for i=1:3
    for j=1:3
        product=i*j;
        fprintf('%d *%d=%d \n', i, j, product);
     end
end
```

Nesting Loops (Cont.)

Result:

```
1*1=1
1*2=3
1*3=3
2*1=2
2*2=4
2*3=6
3*1=3
3*2=6
3*3=9
```

Nesting Loops (Cont.)

- If a *break* or *continue* statement appears inside a set of nested loops, then the statement refers to the *innermost* of the loops containing it.

Nesting Loops (Cont.)

```
for i=1:3
    for j=1:3
        if j==3
           break;
        end
        product=i*j;
        fprintf(‘%d *%d=%d \n’, i, j, product);
     end
     fprintf(‘End of inter loop while i=%d, j=%d\n’,i,j);
end
fprintf(‘End of outer loop while i=%d, j=%d\n’,i,j);
```

Nesting Loops (Cont.)

Result:

```
1 *1=1
1 *2=2
End of inter loop while i=1, j=3
2 *1=2
2 *2=4
End of inter loop while i=2, j=3
3 *1=3
3 *2=6
End of inter loop while i=3, j=3
End of outer loop while i=3, j=3
```

Sincere Thanks!

- Using this group of PPTs, please read
- [1] Yunong Zhang, Weimu Ma, Xiao-Dong Li, Hong-Zhou Tan, Ke Chen, MATLAB Simulink modeling and simulation of LVI-based primal-dual neural network for solving linear and quadratic programs, Neurocomputing 72 (2009) 1679-1687
- [2] Yunong Zhang, Chenfu Yi, Weimu Ma, Simulation and verification of Zhang neural network for online time-varying matrix inversion, Simulation Modelling Practice and Theory 17 (2009) 1603-1617