Exam 1: Lectures 1 – 5, Open-note, JH 245

Sep 23rd (Tue)： 3:30 – 5:30 PM

Lecture 7

Lecture 6: Looping

- **Looping Statements**
  - Allow other statements to be repeated.
- **Two Basic Kinds of Loops**
  - Counted loops: repeat statements a specified number of times
  - Conditional loops: repeat statements, but ahead of time it is not known how many times the statements will be repeated
  - The statements that are repeated in the loop are called the action of the loop.
  - In practice, the **for** statement usually is used as the counted loop, and the **while** statement is used as the conditional loop

Lecture 7: While Loops

➢ While Loops

- The wile statement is used as the conditional loop in MATLAB; It is used to repeat an action when ahead of time it is not known how many times the action will be repeated. The general form of the while statement is:

```
while  condition
    action
end
```

Is it similar to the if statement?

```
if condition
   action
end
```

The action, which consists of any number of statements, is executed as long as the condition is true. The conditions must eventually become false to avoid an infinite loop.

Lecture 7: While Loops

- **While Loops**
  - Example: write a function that will find the first factorial that is greater than the input argument high. factorial(n)=1*2*3*4*…*n
  - Basic idea: Use two variables, one (loop variable) iterates through the integers, one stores the output

factgthigh.m

```
function facgt = factghhigh(high)
% Finds the first factorial > high
i = 0;
fac = 1;
while fac <= high
    i = i +1;
    fac = fac * i;
end
facgt = fac;
```

```
>> factgthigh(5000)
ans =
    5040
```

Can we initialize the loop variable with 1?

Sometimes, we still have a loop variable. This is not a must.

The loop variable and the output variable should be initialized. They are used in the condition and action of while loop.

The action generally includes two parts: calculate the output variable, and increase the loop variable. These two parts also need to be initialized before the while loop.

Lecture 6: Looping

- **Finding Sums and Products**
  - Example:

    Calculate the product of the integers 1 through n: 1 * 2 * 3 * 4 * … * n

    myfact.m

```
function runprod = myfact(n)
% This function returns the product of integers from 1 to n
runprod = 1;
for i = 1:n
    runprod = runprod * i;
end
```

First, the product has to be initialized to one.

Multiply the product with every value

```
>> myfact(5)
ans =
     120
```

The efficient method: MATLAB has a built-in function, **factorial**, that will find the factorial of an integer n.

```
>> factorial(5)
ans =
   120
```

Lecture 7: Looping

- **Multiple Conditions in a While Loops**
  - Example: write a function myanywhile that returns logically true if any value in the input vector is logically true, and logical false otherwise.

myanywhile.m

```
function logresult = myanywhile(vec)
% Uses a while loop so that the action halts as soon as any true value is found
logresult = logical(0);
i = 1;
while i <= length(vec) && logresult == 0
    if vec(i) ~= 0
        logresult = logical(1);
    end
    i = i + 1;
end
```

How to define a logical true and a logical false variable?

An if statement is used as the action in a while loop, to examine if the element is logical true.

```
>> vec = [0  0  0]
>> myanywhile(vec)
```

Can we use vec(i)=1 in the condition?

Lecture 7: Looping

- **Reading from a File in a While Loop**
  - Example: data from an experiment has been recorded in a file called 'experd.dat. The file has some numbers followed by a -99 and then more numbers, all on the same line. The only data values that we are interested in, however, are those before the -99. The algorithm for the script will be:
    - ✓ Read the data from the file into a vector (see load function in Lecture 4)
    - ✓ Create a new vector variable newvec that has the data values only up to but not including the -99
    - ✓ Plot the new vector values, using black o's
  - For example, if the file has the following

    3.1 11 5.2 8.9 -99 4.4 62

Lecture 7: Looping

- **Reading from a File in a While Loop**
  - The programming concept

findvalwhile.m

```
% Reads data from a file, but only plots the numbers
% up to a flag of -99. Uses a while loop.
load experd.dat
i = 1;
while experd(i) ~= -99
    newvec(i) = experd(i);
    i = i + 1;
end
plot(newvec,'ko')
```

Data are stored in a vector with the same name

The newvec is not initialized because it is not used in the condition and action

We did not define newvec ahead of time.
newvec is extended by one element every time.

Lecture 7: Looping

- **Input in a while loop**
  - Example: The following script repeats the process of prompting the user, reading in a positive number, and echo-printing it, as long as the user correctly enters positive numbers when prompted. As soon as the user types in a negative number, the program will print OK and end.

whileposnum.m

```
% Prompts the user and echo prints the numbers entered
% until the user enters a negative number.
inputnum = input('Enter a positive number: ');
while inputnum >= 0
    fprintf('You entered a %d.\n\n', inputnum)
    inputnum = input('Enter a positive number: ');
end
fprintf('OK!\n')
```

```
>> whileposnum
Enter a positive number: 6
You entered a 6.
Enter a positive number: -2
OK!
```

No loop variable is used in this example

The only variable is inputnum.

This statement is used to initialize the expression in the condition

Lecture 7: Looping

- **Counting in a while loop**
  - When it is not known ahead of time how many values will be entered into a script, it is frequently necessary to count the number of values that are entered. For example, if numbers are read into a script, and then the average of the numbers is desired, the script must add them together, and keep track of how many there are, in order to calculate the average.

countposnum.m

```
% Prompts the user for positive numbers and echo prints as
% long as the user enters positive numbers.

% Counts the positive numbers entered by the user
counter = 0;
inputnum = input('Enter a positive number:  ');
while inputnum >= 0
    fprintf('You entered a %d.\n\n', inputnum)
    counter = counter +1;
    inputnum = input('Enter a positive number:  ');
end
fprintf('Thanks, you entered %d positive numbers\n', counter)
```

The counter is similar to the loop variable

```
>> countposnum
Enter a positive number: 4
You entered a 4.
Enter a positive number: -4
Thanks, you entered 1 positive numbers
```

Lecture 7: Looping

- **Error-Checking User Input in a while Loop**
  - In most applications, when the user is prompted to enter something, there is a valid range of values. If the user enters an incorrect value, rather than having the program carry on with an incorrect value, or just printing an error message, the program should repeat the prompt.
  - Example, the following script prompts the user to enter a positive number, and loops to print an error message and repeat the prompt until the user finally enter a positive number.

readonenum.m

```
% Loop until the user enters a positive number
inputnum = input('Enter a positive number: ');
while inputnum < 0
    inputnum = input('Invalid! Entera positive number: ');
end
fprintf('Thanks, you entered a %.1f \n', inputnum)
```

Lecture 7: Looping

- **Symmary for while loop**
  - Format of while loop
  - Multiple conditions in a while loop
  - Reading from a file in a while loop
  - Input in a while loop
  - Counting in a while loop
  - Error checking user input in a while loop

- **Homework on Canvas**