CS100M/CIS121/EAS121
Introduction to Computer Programming

Spring 2004

Lecture 5
MATLAB
Selection Statements

Announcements about Labs

- new lab policies:
  - starting next week, lab exercises will be posted in advance of (or close to) lab time
  - if you can/want to finish on your own, you may do so
  - submitting a completed exercise will indicate your "attendance" in lab
- where do you go?
  - 12:20: if you did NOT get an e-mail about going to Blue Room, go to Upson B7

Announcements about Labs (continued)

- 1:25: Ok to remain: we're going to split you in half. Go to Blue Room first (if there's an overload, a TA will take the overload to the other room)
- 2:30, 3:35: so far you're OK.

Other Announcements

- Partner list
- If you e-mail Kelly, you must indicate M or J!
- Do not respond to e-mail from CMS! Do not e-mail CMS-admin! All e-mail is trashed! (and CMS *does* say *not* to e-mail...)
- A2 due 2/18
  - post questions in News (now that you know how :-)
  - common questions will be answered in A2 FAQ (see link on Assignments page)

Where are we?

- Problems and Solutions
  - Problem Solving Process
  - **algorithms**
- Computer language
  - syntax, semantics
  - character set, **tokens**, **statements**
- "Calculator" statements:
  - **expression**, **function**, **I/O**
- Branching statements
  - **selection** (`if`, `elseif`, `else`, `switch`)
  - **repetition** (`while`, `for`)

Reminder About Logic

- Logical Values
  - true: `1` (actually anything other than `0`)
  - false: `0`
- Relational Operations
  - identity: `==` , `~=`
  - quantity/comparison: `<`, `<=`, `>`, `>=`
  - negation/inverse: `~`
- Logical Operators? (next 2 slides)
- ***Boolean expressions***

Logical Operations

- Logical Operators

and (inclusive)	`&`
and (inclusive, short circuit)	`&&`
or (inclusive)	`|`
or (inclusive, short circuit)	`||`
or (exlcusive)	`xor` (function)
not	`~`

Logical Operations: Examples

```
>> input('Value1: ') | input('Value2: ')

>> input('Value1: ') || input('Value2: ')

>> input('Value1: ') & input('Value2: ')

>> input('Value1: ') && input('Value2: ')

>> xor(1,1), xor(1,0)
```

Selection Statements: Motivation

- Problems:
  - if user enters an illegal value, make them re-enter it
  - if temperature too high, reduce spin
- Some problems have multiple options:
  - if weight too high
    - try eating less
    - or try eating better
    - or try exercising
    - or try doing all of the above
- need ***control structure***:
  - statement that goes to another statement
  - common for languages: choose, repeataf

Selection Statement: Pseudocode

- Want MATLAB to test, then act:
  - if something tested is true, do something(s) then continue with program
  - otherwise skip ahead to rest of program
- Pseudocode:
  - **`if condition is true`**
    `then do these statements`
  - **`otherwise continue with program`**

MATLAB Code

- Syntax

```
if c
  s
  ...
  s
end
```

- Notation
  - *c*: condition, Boolean expression
  - *s*: any MATLAB statement

Example 1

```
clc
disp('Welcome to my wonderful program!')
x = input('Guess a number: ')

if x == 42
   disp('Good guess!')
end

disp('I hope you got it!')
```

Reminders

- `x==42`
  - is a Boolean (logical) expression
  - becomes `0` or `1`
- when program encounters the `if`-statement
  - the program checks the condition
  - if condition is true, then the program does the statements below the condition
  - otherwise, the program continues on
- the program *still* continues on if the condition is true!

Better Style?

- improve commenting:
  - always describe control structure
  - rule of thumb: another programmer should know what it does without looking at its body
- improve behavior:
  - want more options, include "otherwise" behavior
  - use else:
    - if ***c*** is false, MATLAB does statement body of `else`

More Syntax: if-else

- Syntax:

```
if c
  s
  …
  s
else
  s
  …
  s
end
```


Example 2

```
clc
disp('Welcome to my wonderful program!')
guess = input('Guess a number: ')

% Check if GUESS is correct; alert user:
  % good guess:
    if guess == 42
       disp('Good guess!')
  % bad guess:



end

disp('Bye!')
```

Even more syntax!

- What if you want even *more* choices?
- Use **`if-elseif-else`** structure

```
if c1
  statements
elseif c2
  statements
elseif c3
  statements
...
else
  statements
end
```



```
% Set up MATLAB and program:
  clc, clear
  disp('Welcome to my wonderful program!')

% Set up user and computer numbers:
  guess = input('Guess a number: ');
  target = floor(rand*100) + 1; % [1,100]

% Alert user about quality of guess:
  if guess > target
     disp('Too high!')
  elseif guess < target
    disp('Too low!')
  elseif guess == target
    disp('Mazeltov!')
  else
    disp('How did we get here?')
  end

% Wrap things up:
  disp(['Correct guess: ',num2str(target)])
  clear
```


Style & Development

- Indent substructure (the statement body) under a test of a condition (see Slide 17)
- Avoid redundancy! example:
  - if temp > 90 then too hot
  - otherwise if 80 < temp <= 90, just hot
- how to write as code?

Short Circuiting

- use **`&`** and **`|`** to test all expressions in a condition:
  eg) **`if x~=0 & y==1`**
- use **`&&`** and **`||`** to test only the first expression and then the second if necessary:
  eg) **`if x==0 && y==1/x`**
- MATLAB is a bit vague:
  - Helpdesk "?" on if claims that in the context of a selection/repetition statement, **`&`** and **`|`** will also short circuit

Nested Selection Statements

- example)
  if user enters a legal value, test it for another property
- toy example)

```
test1 = 1; test2 = 1; test3 = 0;
if test1
  if test2
    if test3
       disp('Does this display?');
    else
       disp('Or does this?');
    end
  end
end
```


Switch Statements

```
clear, clc
x = input('Enter an integer: ');
switch (x)
    case 0,
        disp('a');
    case {1,2},
        disp('b');
    otherwise,
        disp('c');
end
```


Debugging/Development

- Develop small examples (***toy problems***) to test ideas before you start your main work!
- Output statements
  - display variables and/or messages to help you debug
  - how? ***trace*** the flow of execution

More Debugging/Development

- Test cases
  - determine inputs and expected outputs to program
  - do very simple cases by hand to test algorithm
- Chunkify!
  - develop small chunks of your program from algorithm
  - test each chunk