CS100M/CIS121/EAS121
Introduction to Computer Programming

Spring 2004

Lectures 6, 7
MATLAB Statements
Repetition

Announcements

- Still need a partner? Give note to DIS after lecture (name, Net-ID, 100/121)
- Assignments, Exercise grading info
  - When/where grades posted
  - When/where solutions and GG posted (what's a GG?)
  - Regrade policy reminder
- A2 due next week
- Partners: ***YOU MUST FORM & APPROVE YOUR GROUP ON CMS* BEFORE *THE DUE DATE! (otherwise you will lose style points)***
- Are we OK with labs?

A1 Notes

- Text files and names:
  - For Save As... in Notepad, do not give extension! (it's automatically `.txt`)
  - Wordpad: be sure to save as just text
- Test your code by running your file!!!
  - The program must "compile" in MATLAB (no warnings, no errors) – otherwise, 0 correctness
- Algorithms
  - No code! Must be able to be implemented in any lang.
  - No numbering, unless crucial to access another step
- A.I. reminder

Overview

- New kind of statement: ***repetition***!
- Motivation
- Definite (`for`)
- Tracing
- Indefinite (`while`)
- Problem Solving Process revisited
- Continued in Part 2

Motivation

- How to find mean? (eg, average grade)
- How to find a max/min grade?
- How to find the root of an equation? (A3)
- Ideas:
  - Accumulating data
  - Searching data
  - Repeating instructions

Example: Statistics

- Use MATLAB for average: `mean`
  - See `help mean` and `type mean`
- No MATLAB? "Under the Covers?"
  Here's an algorithm:

```
total←0, count←0
Get a legal grade
If not last grade
  increment count
  add grade to total
  get next grade
Repeat
Otherwise, report total/count
```


Language Reminder: Assignment

- How do you "increment the count?"
- Assignment:
  `var = expression`
- What if ***`expression`*** contains ***`var`***?

```
x = 1;
x = x + 1;
```

- Assignment (effectively) works right to left:
  - first `x + 1` is evaluated using current value of `x`
  - then, `x` gets result of 2.

Using Selection (bad...why?)

```
clc,clear
total = 0;
count = 0;

grade = input('Enter 1st grade: ');
count = count + 1;
total = total + grade;

grade = input('Enter 2nd grade: ');
count = count + 1;
total = total + grade;

disp(['Mean: ',num2str(total/count)]);
```

How to improve?

- Need new kind of control structure
  - want to repeat statements without *rewriting* statements
  - why? ***redundant code*** makes program too long!
  - control structures execute other statements...
- Counting loop:

```
set up counter
if count not equal to counter
   do something(s)
   increment counter
repeat
```

- Why called ***loop***?
- Shorter version: "for each step, do stuff"

MATLAB `for`-loops

- Syntax:

```
for count = array
  statements
end
```

- example)

```
for ii = 1:10  % why ii?
  disp('hi');
end
```

- example)

```
for ii = 10:-1:1
  ii
end
```


Tracing `for`-loops

- How to say?
  "For a **COUNT** from **HERE** to **THERE** by an **INCREMENT**, do **STATEMENTS**."
- ***Tracing*** for debugging:
  - display a counter in the statements
  - print out current values at each step of counter
  - hint: set ***break points*** in code

Designing `for`-loops

- `for` loop use:
  - used for ***definite loop***
  - goes *certain* amount of steps (also called ***iterations***)
- Design:
  - Problem must involve a known amount of iterations
  - Figure out how to handle just **one** iteration before writing the full loop

Application

```
% Better mean program (goodmean.m)

% Initialize data:
clear, clc
STUDENTS = input('Enter number of students: ');
grade = -1; % current grade entered so far
total = 0; % total of all grades so far

% For each student, find the current grade and
% add to total:
for count = __________
   ?                        % get next grade

   ?                        % increment total

end

% Report result:
disp(['Mean grade: ',num2str(total/STUDENTS)]);
```


Fun With MATLAB!

- How to irritate everyone in lab:
  `for ii=1:100,beep,pause(0.1),end`
- Unrelated to loops, but related to beeps:
  - see `wavplay`
  - example? download `music.m` from lecture notes (someone ask me about posted files and format)

General Loops

- What if you need to find something in array?
- example) Find a random value in an array 1:10
- Using `for`, but bad style:

```
data = 1:10;
target = floor(10*rand+1);
for ii=1:length(data)
  if target==data(ii)
    disp('success!');
                % need to put something here
  end
end
```


Improving Style

- Do not use `break`!
  - in small programs usually OK, but not needed
  - if loop becomes very nested, very difficult to trace during debugging and review
- Need more general form:
  while a Boolean **expression** remains true
  do **statements**
  continue with program if **expression** became/is false

MATLAB **while**-loop

- Syntax:

```
while expression
  statements
end
```

- example) Count from 1 to 10!

```
count = 1;
while count < 11   % should we say count <= 10 ?
   count = count + 1
end
```

- Note similarity to **for**...let's summarize the patterns we know up to this point...

Patterns for Definite Loops

- **for**:

```
for counter=init:incr:final
  statements
end
```

- **while**:

```
counter=init;
while counter <= final
  statements
  counter=counter+incr
end
```


Why **while** is useful...

- prompting:

```
val = input('Enter a valid grade: ');
while val < 0 || val > 100
   disp('Pay attention!');
   val = input('Enter a valid grade: ');
end
```

- general case:
  - not sure how much data/many steps to check
  - called ***indefinite loop***: could actually go on forever!
- dangerous!

```
while true, stuff, end;
```


Application: Max Grade

- Some design:
  - get input from user
  - pick out *legal* max value
- algorithm:

set initial **maxgrade** to out-of-bounds value
get a **grade**
if grade **is legal**
    if **grade exceeds maxgrade**
        grade becomes new **maxgrade**
    get **next grade** to process
repeat
report results
    if **maxgrade** is **out-of-bounds**, nothing legal was entered
    otherwise, report **maxgrade**

```
% Initialize data:
clc, clear       % setup MATLAB
STOP = -1;       % stopping value for grade entry
maxGrade = STOP; % max grade so far
grade = input('Enter a grade: '); % grade so far

% Find max grade from user input until out-of-bounds
% value is entered:
while grade >= 0 && grade <= 100

   % Update max grade if current grade is bigger:
   if grade > maxGrade, maxGrade = grade; end

   % Get next grade to process:
   grade = input('Enter a grade: ');

end

% Report results:
if maxGrade == STOP
    disp('nothing valid was entered')
else
    disp(['Max grade: ',num2str(maxGrade)])
end
```


Style Issues

- use of named constants: eg, `STOP`
- comments
  - for blocks of code and control structures
  - go above code blocks or to right of ***single*** line
- indentation of substructures
- indefinite loop, so used `while`
- the good, the bad, and the ugly....
  - some suggestions to improve?

Designing Indefinite Loop

- Elements from programming process:
  - identify your nouns!
  - identify specifications!
  - develop algorithm!
- 3 “parts”:
  - initialize variables (see above)
  - loop
  - post-loop analysis

Application: Number Guessing Game

- Computer picks a random integer.
- Prompts user to guess.
- Limits amount of user's guesses to reasonable #
- Allows user to quit game.
- Other client specifications?

Variable Initialization

- Set constants
- Set variables to current values
  - possibly out-of-bound values
  - think of variables as what you know "so far" (pre-loop)
- Get initial values from user if necessary
- Number guessing:
  - need limits, target, guess, count, way to quit

Design of Loop

- What must be true at all times during the loop?
  - think in terms of *i*th step of loop
  - forms loop condition
- Pattern:
  set/get values (see Initialization)
  if conditions are still true
  account for the values you sought/got
  update the variables
  get next value(s) to test
  repeat

Number Guessing

- user makes guess
- check: guess OK? too many guesses? stop value?
- if OK:
  - count the guess
  - check if guess too high/too low
  - get next guess
- repeat

Post-Loop (Post Processing)

- what happens if the loop was not entered? (good use for boundary values!)
- what if happens if target is found?
- Number guessing:
  - user quit early
  - user took too long
  - user got it right!

Number Guessing Code

```
clc, clear          % clean things up
% lowGuess:         % lowest value in range to guess
% highGuess:        % highest value in range to guess
% guess:            % value that user guesses
% target:           % value that computer picks
MIN = -1000;        % smallest value that user may pick
MAX = 1000;         % largest value that user may pick
name  = 'ng';       % name of the script
count = 0;          % # of valid guesses from user

% more stuff
```



```
high = readInt(MIN, MAX,' Enter the highest value: ');
low  = readInt(MIN,high,' Enter the lowest value:  ');

target = ceil(rand*(high-low)) + low;

guess = readInt(low-1,high+1,' Guess a number: ');

while guess ~= target && ...                          guess <= high   && ...
    guess >= low    && ...
    count <= high-low

  count = count+1;

  if guess < target                                   disp(['  ',num2str(guess),' is too low!']);
  elseif guess > target                               disp(['  ',num2str(guess),' is too high!']);

  end

  guess = readInt(low-1,high+1,' Guess a number: ');

end   % but still more...
```



```
if guess==target
   disp([' Congratulations!']);
elseif guess > high | guess < low
   disp('  Halting!');
   disp(['  The correct guess was ',num2str(target),'.']);
elseif count > high-low
   disp('You were taking too long, so I quit.');
else
   disp('Something horrible happened. Fix the program!');
end

% see on-line example for more complete version!
```


Conditional Update Pattern

```
max = -1;
v = input('Get value: ');
while v ~= STOP
  if v > max, max = v; end;
  v = input('Get value: ');
end
if max == -1
  disp('no values');
else
  max
end
```