Week 7 - Programming II

- Today – more features:
  - Loop control
  - Extending if/else
  - Nesting of loops
- Debugging tools
- Textbook chapter 7, pages 190-192 (sections 7.2.3, 7.5, 7.6)

Loop Controls

Loops contain sets of commands that you want to do repeatedly.
You might want to:
- Skip commands in the current iteration
- Stop the loop itself

Why continue once you’ve found what you’re looking for!!

Skipping Ahead: Continue

Continue – jumps to next loop iteration:

```matlab
for k = 1:25000
  if x(k) > 0
    continue
  end
  { more commands }
end
```

Early Termination: Break

Break – ends the loop:

```matlab
for variable = {array of length n}
  ..... break
end
goto commands beyond end
```

Example – calculating interest until the amount doubles using a for loop:

```matlab
value = 1000;
for year = 1:1000
  value = value * 1.08;
  disp(['$',num2str(value), ' after ', num2str(year), ' years'])
  if value >= 2000
    break
  end
end
```

Command Window

- $1080.00 after 1 years
- $1166.40 after 2 years
- $1259.71 after 3 years
- $1360.49 after 4 years
- $1469.33 after 5 years
- $1596.87 after 6 years
- $1743.82 after 7 years
- $1905.00 after 8 years
- $2186.92 after 9 years
- $2408.62 after 10 years

only needed 10 years
Example – accept input, appending it to vector, until a negative number is entered:

```matlab
vector = []; allow up to 1000 values, if necessary
for count = 1:1000
    value = input('next number: '); if value < 0 break
    else vector(end+1) = value; end
end vector
```

Extensions of if/else

As introduced, if/else allows for two choices:

```matlab
if expression
    (commands if expression is true )
else
    (commands if false )
end
```

What if there are more than 2 situations?

3 situations:

find the largest of 3 variables a, b, c

```matlab
a \geq b \geq c a \geq c \geq b
b \geq a \geq c b \geq c \geq a
c \geq b \geq a c \geq a \geq b
```

4 situations:

convert a compass angle to a direction:

```matlab
0^\circ \rightarrow \text{east} 90^\circ \rightarrow \text{north} 180^\circ \rightarrow \text{west} 270^\circ \rightarrow \text{south}
```

Could use “nested” if/else commands

```matlab
% find the largest of a, b, and c
if a \geq b & a \geq c
    result = a;
else
    if b \geq c
        result = b;
    else
        result = c;
    end
end
```
The "elseif" command

if expression1
    {commands if expression1 is true}
elseif expression2
    {commands if expression2 is true}
else
    {commands if both expressions are false}
end

Examples:

if a >= b & a >= c
    result = a;
elseif b >= c
    result = b;
else
    result = c;
end

Note – many elifs are allowed, but only 1 "else"

Examples – Hi-Lo: a guessing game with feedback

select hidden number

input guess

correct?

yes

no

provide hi/lo feedback

5 tries?

yes

lose

no

win

Loops within Loops – Nesting

for index1 = array1
    {outer loop commands}
    for index2 = array2
        {inner loop commands}
    end
    {more outer loop commands}
end
can be more than 2 levels deep

Variable values by example

for index1 = 1:4
    for index2 = [3 6 9]
        {index1 index2}
    end
end
Example – computing a table of \( z = x^2 + y^2 \) for \( x \) and \( y \) equal to the integers 1, 2, ..., 6:

```matlab
for x = 1:6
    for y = 1:6
        z(x,y) = x^2 + y^2;
    end
end
```

<table>
<thead>
<tr>
<th>( x )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>17</td>
<td>26</td>
<td>37</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>10</td>
<td>18</td>
<td>29</td>
<td>41</td>
<td>55</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>29</td>
<td>41</td>
<td>66</td>
<td>90</td>
<td>125</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>29</td>
<td>66</td>
<td>109</td>
<td>161</td>
<td>230</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>41</td>
<td>90</td>
<td>161</td>
<td>250</td>
<td>365</td>
</tr>
<tr>
<td>6</td>
<td>37</td>
<td>55</td>
<td>125</td>
<td>230</td>
<td>365</td>
<td>530</td>
</tr>
</tbody>
</table>

Solution – use nested loops to try all combinations, skipping repeats:

First, let's initialize variables:

```matlab
skills = [7 4 4 2; 6 8 2; 4 1 3; 6 5 1];
best_so_far = -Inf;
```

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Sue</td>
<td>6</td>
<td>8</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Bob</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Liz</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Test a valid assignment for quality:

```matlab
quality = skills(1,Joe) + skills(2,Sue) + ...
         skills(3,Bob) + skills(4,Liz);
if quality > best_so_far
    best_assignment = [Joe, Sue, Bob, Liz];
    best_so_far = quality;
end
```

And then terminate the 4 for loops:

```matlab
end
end
end
```

Example – matching of people's skills and tasks:

Situation:
- 4 tasks
- 4 people with different skills to do them

Skill table as shown:

Goal – assign tasks to maximize the sum

Example solution of 20

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Sue</td>
<td>6</td>
<td>8</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Bob</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Liz</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Next, start nested loops:

```matlab
for Joe = 1:4
    for Sue = 1:4
        for Bob = 1:4
            for Liz = 1:4
                if Sue == Joe
                    continue
                end
                if Bob == Sue || Liz == Sue
                    continue
                end
                if Liz == Joe || Liz == Sue || Liz == Bob
                    continue
                end
                Test a valid assignment for quality:
                quality = skills(1,Joe) + skills(2,Sue) + ...
                skills(3,Bob) + skills(4,Liz);
                if quality > best_so_far
                    best_assignment = [Joe, Sue, Bob, Liz];
                    best_so_far = quality;
                end
            end
        end
    end
end
```

The result:

```matlab
best_assignment = [3 2 4 1]
best_so_far = 21
```
Debugging Tools

- Debugging = finding and correcting errors (bugs) in programs

- Useful debugging tools:
  - Ability to stop a program in the middle of its execution (at a breakpoint)
  - Ability to examine variable values at that point
  - Ability to modify variable values at that point

What shows up at the breakpoint

- Command window:
- Editor window:

Can single step (F10) or continue (F5) to the next breakpoint (or end)