Statements We Will Use in Flow Control

- **for loops**
  - repeat a set of commands for a specified number of times

- **while loops**
  - repeat a set of commands while a specified condition is true

- **if, else, elseif**
  - execute a set of commands if a specified condition is true (if) or execute an alternative set of commands if the specified condition is false (else, elseif)

### while loop

- Syntax:
  ```matlab
  while <a specified condition is true>
  Matlab commands
  This may be several lines
  Repeat this set of commands until the condition is no longer true
  end
  ```

- Usually used when you do NOT know how many times you may need to repeat the given Matlab commands.

### while loop continued

- **Step 1:** Identify the specified condition (test) that must be TRUE to STOP the operation.

- **Step 2:** Use the OPPOSITE form of the test in Step 1 as the while loop condition statement.

- **Step 3:** Set up a way of changing (incrementing) the test condition INSIDE the while loop so that the loop will eventually end. Set up calculations.

- **Step 4:** BEFORE the while loop, set up (initialize) all necessary variables so the test condition and all calculations will evaluate properly the first time.

### while loop example 1

- Repeat for loop example 1 (summation of the first 10 integers) using a while loop.

- We want to add $1 + 2 + 3 \ldots + 10$ so we want to STOP when the value of our index (integer) is 11.

- Therefore, our test condition is:
  ```matlab
  index variable <= 10
  ```

  That is, as long as the current integer is less than or equal to 10 keep adding.
**while loop** example 1 continued

- A solution, lines 8 - 13:

```matlab
i_sum = 0       % Initialize summation variable
ii = 1          % Initialize index
while ii <= 10  % Check if condition is true, are we still in the 1 to 10 range
    i_sum = i_sum + ii % If condition is true, add to sum
    ii = ii + 1 % If condition is true, increment index
end % Loop back up and check condition again
```

Remove the semicolons save and run.

---

**while loop** example 1 continued

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Initial ii</th>
<th>i_sum</th>
<th>Updated ii</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0 + 1 = 1</td>
<td>1 + 1 = 2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1 + 2 = 3</td>
<td>2 + 1 = 3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3 + 3 = 6</td>
<td>3 + 1 = 4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>6 + 4 = 10</td>
<td>4 + 1 = 5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10 + 5 = 15</td>
<td>5 + 1 = 6</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>15 + 6 = 21</td>
<td>6 + 1 = 7</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>21 + 7 = 28</td>
<td>7 + 1 = 8</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>28 + 8 = 34</td>
<td>8 + 1 = 9</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>34 + 9 = 45</td>
<td>9 + 1 = 10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>45 + 10 = 55</td>
<td>10 + 1 = 11</td>
</tr>
</tbody>
</table>

---

**while loop** example 2

- Create a list of the powers of 3 less than or equal to 1000, i.e., create the list:

```matlab
pow_of_3 = [3 9 27 81 ... ]
```

- Our STOP condition is when $3^N$ is greater than 1000. Therefore our *while loop* test condition is $3^N \leq 1000$.

**while loop** example 2 continued

- A solution, lines 16 - 20:

```matlab
ii = 1          % Initialize index to 1st power
while 3.^ii <= 1000 % Check to see if 3 to the current power is less than 1000
    pow_of_3(ii) = 3.^ii % If true, calculate 3 to the current power and store in proper location
    ii = ii + 1 % If true, increment index to next power
end % Loop back up and check condition again
```

Remove the semicolons save and run.
**while loop** example 2 continued

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Ini ii</th>
<th>Test pow_of_3</th>
<th>Updated ii</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3(^1) = 3</td>
<td>1 + 1 = 2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3(^2) = 9</td>
<td>2 + 1 = 3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3(^3) = 27</td>
<td>3 + 1 = 4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3(^4) = 81</td>
<td>4 + 1 = 5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3(^5) = 243</td>
<td>5 + 1 = 6</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>3(^6) = 729</td>
<td>6 + 1 = 7</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>3(^7) = 2187</td>
<td>STOP</td>
</tr>
</tbody>
</table>

What is the value of \( ii \) after the **while loop** is finished? Why?

What is the highest power \( N \) that satisfies our condition? Are the values of \( N \) and \( ii \) equal? Why or why not?

---

**while loop** example 3

- Assume we have the following lists of time and their associated velocities:
  
  \[
  t = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7] \\
  v = [10\ 12\ 14\ 9\ 3\ -2\ -7\ -10]
  \]

- Find the time that the velocity first goes negative and find the associated velocity.
- By inspection \( t = 5 \), \( v = -2 \).

---

**while loop** example 3 continued

- A solution, lines 23 - 30:

  \[
  t = \[0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\] \% Initialize \\
  v = \[10\ 12\ 14\ 9\ 3\ -2\ -7\ -10\] \% \( t \), \( v \), and \( ii \) \% an index. \\
  ii = 1 \% Check current \( v \) \% to see if is \% non-negative. \\
  while \( v(ii) >= 0 \) \% If the test is true, \% increment the index. \\
      ii = ii + 1 \% End while loop. \\
  end \\
  t_neg = t(ii) \% Get values of \( t \) \% and \( v \).
  v_neg = v(ii)
  
  Remove the semicolons save and run.

---

**while loop** example 3 continued

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Initial ii</th>
<th>v(ii)</th>
<th>Updated ii</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>v(1) = 10</td>
<td>1 + 1 = 2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>v(2) = 12</td>
<td>2 + 1 = 3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>v(3) = 14</td>
<td>3 + 1 = 4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>v(4) = 9</td>
<td>4 + 1 = 5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>v(5) = 3</td>
<td>5 + 1 = 6</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>v(6) = -2</td>
<td>STOP</td>
</tr>
</tbody>
</table>

What is the value of \( ii \) after the **while loop** is finished? Why?

Why do we use the current value of \( ii \) after the **while loop** is finished to find our desired values?
Error Checking

- A common use of Flow Control is "Error Checking."
- We can check for input errors, calculation errors, data out-of-range errors, etc.
- Often requires "nested" Flow Control Structures.

- **Example:** We will check to make sure the user “inputs” the number 1 or 2 or 3 only.
- Introduces the **input** statement
  - Allows user input from the keyboard.
- Introduces the **fprintf** statement.
  - We will cover **fprintf** in lab 5.3.
- Remove the $ sign from the beginning of lines 33-49.
  - Input obvious errors and see what happens.
  - Input one of the three “correct” values and see what happens.

In-Class Problem

- Create a new function **vecthreshold**
  - finds the value and location (or address) of the first term in a vector that exceeds a given threshold.

- Modify our force-plate **m.file** so that the program:
  - uses the function **vecthreshold** to determine the time when $F_x$, $F_y$, and $F_z$ first exceed 100 N.

- Everyone should have completed **m.files** by the end of the period.

- The completed **m.files** will include proper documentation.

Our Plan for the Function **vecthreshold**

```matlab
function [tval, tid] = vecthreshold(f, thresh)
% vecthreshold - finds the first value in a vector of numbers
% that exceeds a specified threshold value
% Usage: [tval, tid] = vecthreshold(f, thresh)
% Input: f - a vector of numbers
% thresh - the threshold value
% Output: tval - the value in f that first exceeded the threshold value
% equals -1 if no values in f exceeds the threshold
% tid - the location address (index) of the the first value in f that exceeded the threshold
% equals 0 if no values in f exceed the threshold
% Need real Matlab stuff here.

return
```

Our Plan for the Force-Plate Program