Branches

Branches are used to select and execute specific sections of the code while skipping other sections.

Selection of different sections depend on a condition statement.

We will learn:
- if statement
- switch statement

Branching Examples

- Examples:
  - if ( r <= 0 ),
    disp( ['Radius must be positive']);
  end

  - if ( grade < 0 ) | ( grade > 100 ),
    disp( ['Grade must be in [0,100] range']);
  end

  - if isninf( result ),
    disp( 'Result is infinite');
  end
  
- Water tank example:
  
r = input('Enter the radius of the tank base (in meters):');
if r <= 0
    error( 'Radius must be positive');
end

h = input('Enter the height of the tank (in meters):');
if h <= 0
    error( 'Height must be positive');
end

w = input('Enter the amount of water (in m3):');
if w <= 0
    error( 'Amount of water must be positive');
end

capacity = pi * r^2 * h;

space = capacity - w;
if space > 0
    disp( ['There is ' num2str(space) ' m3 extra space']);
else
    disp( 'Tank is full');
end
Branches: “if-else” Statement

```
if ( condition ),
  statement 1
  statement 2
  ...
else
  statement 1
  statement 2
  ...
end
```

Branching Examples

Example: Assigning letter grades

<table>
<thead>
<tr>
<th>Range</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 ≥ grade &gt; 95</td>
<td>A</td>
</tr>
<tr>
<td>95 ≥ grade &gt; 86</td>
<td>B</td>
</tr>
<tr>
<td>86 ≥ grade &gt; 76</td>
<td>C</td>
</tr>
<tr>
<td>76 ≥ grade &gt; 66</td>
<td>D</td>
</tr>
<tr>
<td>66 ≥ grade &gt; 0</td>
<td>F</td>
</tr>
</tbody>
</table>

How can we compute the letter corresponding to a given numeric grade?

Branching Examples

```
if ( grade > 95 ),
  disp( 'Grade is A' );
else
  if ( grade > 86 ),
    disp( 'Grade is B' );
  elseif ( grade > 76 ),
    disp( 'Grade is C' );
  elseif ( grade > 66 ),
    disp( 'Grade is D' );
  else
    disp( 'Grade is F' );
  end
end
```

Branches: “if-elseif-else” Statement

```
if ( condition 1 ),
  statement 1
  statement 2
  ...
elseif ( condition 2 ),
  statement 1
  statement 2
  ...
else
  statement 1
  statement 2
  ...
end
```

Branching Examples

Example: Finding roots of the quadratic equation “ax^2 + bx + c = 0”

Pseudocode:

- d = b^2 - 4ac
- if d > 0,
  two real roots
  else if d == 0,
    two identical roots
  else
    two complex roots
Branching Examples

% Prompt the user for the coefficients of the equation
a = input('Enter the coefficient A: ');
b = input('Enter the coefficient B: ');
c = input('Enter the coefficient C: ');
% Calculate discriminant
discriminant = b^2 - 4*a*c;
% Solve for the roots, depending on the value of the discriminant
if discriminant > 0 % there are two real roots, so...
    x1 = (-b + sqrt(discriminant)) / (2*a);
    x2 = (-b - sqrt(discriminant)) / (2*a);
    disp('This equation has two real roots:
         x1 = %f
         x2 = %f', x1, x2);
elseif discriminant == 0 % there is one repeated root, so...
    x1 = -b / (2*a);
    disp('This equation has two identical real roots:
         x1 = x2 = %f', x1);
else % there are complex roots, so ...
    real_part = -b / (2*a);
    imag_part = sqrt(abs(discriminant)) / (2*a);
    disp('This equation has complex roots:
         x1 = %f +i %f
         x1 = %f -i %f', real_part, imag_part, real_part, imag_part);
end

example: Decision for playing tennis

outlook = input('How is the outlook? (o)vercast, (s)unny, (r)ainy: ', 's');
if (outlook == 'o'),
    disp('You can play tennis');
elseif (outlook == 's'),
    humidity = input('How is humidity? (h)igh, (l)ow: ', 's');
    if (humidity == 'h'),
        disp('I do not recommend you play tennis');
    elseif (humidity == 'l'),
        disp('You can play tennis');
    else
        disp('Invalid humidity info');
    end
elseif (outlook == 'r'),
    wind = input('How is the wind? (s)trong, (w)eak: ', 's');
    if (wind == 's'),
        disp('I do not recommend you play tennis');
    elseif (wind == 'w'),
        disp('You can play tennis');
    else
        disp('Invalid wind info');
    end
else
    disp('Invalid outlook info');
end

Branches: “switch” Statement

switch (expression),
    case value 1,
        statement 1
    case value 2,
        statement 2
    default,
        statement 1
    case {value set 1},
        statement 1
    case {value set 2},
        statement 2
    otherwise,
        statement 1
end
Branching Examples

- Example: Odd or even numbers
  ```matlab
  switch (value),
  case {1,3,5,7,9},
      disp( 'Odd number' );
  case {2,4,6,8,10},
      disp( 'Even number' );
  otherwise,
      disp( 'Out of range' );
  end
  ```

Branching Examples

- Example: Unit converter
  ```matlab
  x = input( 'length (in cm): ' );
  u = input( 'unit: ', 's' );
  switch (u),
  case { 'cm', 'centimeter' },
      disp( [ num2str(x) 'cm' ] );
  case { 'mm', 'millimeter' },
      disp( [ num2str(10*x) 'mm' ] );
  case { 'm', 'meter' },
      disp( [ num2str(x/100) 'm' ] );
  case { 'in', 'inch' },
      disp( [ num2str(2.54*x) 'in' ] );
  otherwise,
      disp( 'Unknown unit' );
  end
  ```

Loops

- Loops are used to execute a sequence of statements more than once
- We will learn:
  - `while` loop
  - `for` loop
- They differ in how the repetition is controlled

Loops: "while" Loop

- Statements are executed indefinitely as long as the condition is satisfied

```
while ( condition ),
  statement 1
  statement 2
  ...
end
```

Loop Examples

- Example: Arithmetic mean and standard deviation of non-negative measurements
- Pseudocode:
  ```plaintext
  Initialize sum_x, sum_x2, n
  Read first value, x
  while x >= 0,
      n = n + 1;
      sum_x = sum_x + x;
      sum_x2 = sum_x2 + x^2
  Read next value, x
  end
  x_mean = sum_x / n;
  std_dev = sqrt( ( n * sum_x2 – sum_x^2 ) / ( n * (n-1)) );
  Display results to the user
  ```

Loop Examples

```
% Initialize sums.
n = 0; sum_x = 0; sum_x2 = 0;
% Read in first value
x = input('Enter first value: ');
% While Loop to read input values.
while x >= 0
    n = n + 1;
    sum_x = sum_x + x;
    sum_x2 = sum_x2 + x^2;
    x = input('Enter next value: ');
end
% Calculate the mean and standard deviation
x_bar = sum_x / n;
std_dev = sqrt( (n * sum_x2 – sum_x^2) / (n * (n-1)) );
% Tell user.
fprintf('The mean of this data set is: %f
', x_bar);
fprintf('The standard deviation is: %f
', std_dev);
fprintf('The number of data points is: %f
', n);
```
Loops: “for” Loop

- Statements are executed a specified number of times

for index = expression,
   statement 1
   statement 2
... 
end

- Expression is usually a vector in shortcut notation first:increment:last

Loop Examples

Example:

```
for x = 1:2:10,
   x
end
```

Output:

```
x =
1
3
5
7
9
```

Example:

```
for x = [1 5 13],
   x
end
```

Output:

```
x =
1
5
13
```

Example:

```
for x = [1 2 3; 4 5 6],
   x
end
```

Output:

```
x =
1
4
2
5
3
6
```

Example: Factorial (n!) of an integer n

```
n = input( 'Please enter n: ' );
if ( ( n < 0 ) | ( fix(n) ~= n ) ),
   error( 'n must be a non-negative integer' );
end
if ( ( n == 0 ) | ( n == 1 ) ),
   f = 1;
else
   f = 1;
   for ii = 2:n,
      f = f * ii;
   end
end
```

Example: Arithmetic mean and standard deviation of non-negative measurements

Pseudocode:

```
Initialize sum_x, sum_x2
Read the number of measurements, n
for i = 1:n,
   Read value, x
   sum_x ← sum_x + x
   sum_x2 ← sum_x2 + x^2
end
x_mean ← sum_x / n
x_std ← sqrt( ( n * sum_x2 - sum_x^2 ) / ( n * (n-1) ) )
Display results to the user
```
Loop Examples

% Initialize sums.
sum_x = 0; sum_x2 = 0;
% Get the number of points to input.
% Input format: number of points
if (n < 2) % Insufficient data
    disp('At least 2 values must be entered.');
else
    % we will have enough data, so let's go on.
    % Loop to read input values.
    for ii = 1:n
        x = input('Enter value: ');
        sum_x = sum_x + x;
        sum_x2 = sum_x2 + x^2;
    end
    % Now calculate statistics.
    x_bar = sum_x / n;
    std_dev = sqrt((n * sum_x2 - sum_x^2) / (n * (n - 1)));
    % Tell user.
    fprintf('The mean of this data set is: %f
', x_bar);
    fprintf('The standard deviation is:    %f
', std_dev);
    fprintf('The number of data points is: %f
', n);
end

Loop Examples

% Example: Guessing a number computer picks between 1 and 10
% Pseudocode:
% Pick a random number, num, in [1,10]
% Read user's guess
% while ( guess ~= num ),
% Read user's new guess
end
num = round( (10-1) * rand + 1 );
guess = input( 'Your guess?' );
tries = 1;
while ( guess ~= num ),
    guess = input( 'Your guess?' );
    tries = tries + 1;
end
if ( guess == num ),
    disp( 'Congratulations!' );
else
    disp( 'You could not guess correctly' );
end

Loop Examples

% Example: Nested loops
for ii = 1:3,
    for jj = 1:5,
        p = ii * jj;
        fprintf( '%d x %d = %d
', ii, jj, p );
    end
end
Loops: “break/continue” Statements

- **Break** statement terminates the execution of a loop and passes the control to the next statement after the end of the loop.
- **Continue** statement terminates the current pass through the loop and returns control to the top of the loop.

Loop Examples

- **Example:**
  
  ```matlab
  for ii = 1:5,
    if ( ii == 3 ),
      break;
    end
    fprintf( 'ii = %d
', ii );
  end
  disp( 'End of loop' );
  
  **Output:**
  
  ii = 1
  ii = 2
  End of loop
  ``

- **Example:**
  
  ```matlab
  for ii = 1:5,
    if ( ii == 3 ),
      continue;
    end
    fprintf( 'ii = %d
', ii );
  end
  disp( 'End of loop' );
  
  **Output:**
  
  ii = 1
  ii = 2
  ii = 4
  ii = 5
  End of loop
  ``

- **Number guessing example: User has only 3 tries**

  - **Pseudocode:**
    - Pick a random number, num, in [1,10]
    - for tries = 1:3,
      - Read user’s new guess
      - Stop if guess is correct
    - end

  ```matlab
  num = round( (10-1) * rand + 1 );
  for tries = 1:3,
    guess = input( 'Your guess?' );
    if ( guess == num ),
      disp( 'Congratulations!' );
      break;
    end
    disp( 'You could not guess correctly' );
  end
  ```

Advice

- Use indentation to improve the readability of your code.
- Never modify the value of a loop index inside the loop.
- Allocate all arrays used in a loop before executing the loop.
- If it is possible to implement a calculation either with a loop or using vectors, always use vectors.
- Use built-in MATLAB functions as much as possible instead of reimplementing them.