Topics: Iteration using for, while

Reading CFile: Sec 2.1
     KU: Sec 4.5

Iteration

Important features:

• Task can be accomplished if some step is repeated a number of times
• Must have a starting point
• Must know when to stop ⇒
• Must keep track of progress ⇒

Example 1: Area of $n$-gon

Complete the following program to compute and display the areas of inscribed and circumscribed regular $n$-gons in the unit circle where $n = L, L + 1, \ldots, R - 1, R$. Use only scalar variables.

```matlab
L = input('Enter lower bound for n: ');
R = input('Enter upper bound for n: ');
fprintf('
 n	Inner Area	Outer Area
');

% Compute and display areas of n-gons
innerA = (n/2)*sin(2*pi/n);
outerA = n*tan(pi/n);
fprintf('%d	%.6f	%.6f
', n, innerA, outerA);
```

Syntax of the for Loop

```
for <index variable> = <lower bound>: <increment>: <upper bound>
    Statements to execute
    Also called loop body
end
```

The index variable takes on the values specified in the loop header one at a time, i.e., one value for each pass through the loop.
Example 2: “Accumulating” a solution using iteration

Do you need to initialize variables? Consider variable total in the following program:

```matlab
% Average 10 numbers from user input
n = 10; % number of data values
total = 0; % current sum (initialized to zero)
for k = 1:n
    % read and process input value
    num = input('Enter a number: ');
    total = total + num;
end
ave = total/n % average of n numbers
```

Example 3: \( \text{n-gon} \rightarrow \text{circle} \)

As \( n \) increases, the regular inscribed and circumscribed \( n \)-gons converge to the circle. Since the area of the unit circle is \( \pi \), we have

\[
\lim_{n \to \infty} \text{innerArea}_n = \pi \quad \lim_{n \to \infty} \text{outerArea}_n = \pi.
\]

Write a program to find \( n \) “sufficiently large” to approximate the area of the unit circle.

Algorithm:

1. \( n \) gets initial value
2. Repeat until tolerance is reached:
   - Calculate innerA, outerA for current \( n \)
   - \( \text{diff} = \text{outerA} - \text{innerA} \)
   - Increase \( n \)

Program:

```matlab
fprintf('
 n	 A(n)	 B(n)
');
% Initialization
n = 3;
while ( )
    fprintf('%4d %9.6f %9.6f 
', n, innerA, outerA);
    n = n+1;
    innerA = (n/2)*sin(2*pi/n);
    outerA = n*sin(pi/n)/cos(pi/n);
end
```
Syntax of the while Loop

```python
while condition
    statements to execute if expression evaluates to true
end
```

If the condition evaluates to true, the loop body executes. Then the condition is evaluated again. When the condition evaluates to false, the loop body is skipped and the program continues after the `end` keyword.

Two useful patterns

<table>
<thead>
<tr>
<th>Pattern for doing something ( n ) times</th>
<th>Pattern for doing something an indefinite number of times</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>for</strong> ( k = 1:n )</td>
<td>% initialization % ... % do something % do something % update status (variables) % ... end</td>
</tr>
<tr>
<td>% ...</td>
<td>end</td>
</tr>
</tbody>
</table>

Brute-force algorithm to find minimum function value

How do we find the minimum value of some function \( f(x) \)?

- Start at \( x = L \)