Topics: Iteration using while

Reading (ML): Sec 4.1

Iteration

Important features:

- Task can be accomplished if some step is repeated a number of times
- Must be able to quantify success ⇒
- Must have a starting point
- Must keep track of progress ⇒

Syntax of the while Loop

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

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>( k = 1; )</td>
<td>( % ) initialization</td>
</tr>
<tr>
<td>while ( k \leq n )</td>
<td>( % ) ...</td>
</tr>
<tr>
<td>% do something</td>
<td>while ( ) not stopping signal</td>
</tr>
<tr>
<td>% ...</td>
<td>% do something</td>
</tr>
<tr>
<td>( k = k + 1; )</td>
<td>% ...</td>
</tr>
<tr>
<td>end</td>
<td>% update status (variables)</td>
</tr>
<tr>
<td></td>
<td>% ...</td>
</tr>
</tbody>
</table>

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.

```
L = input('Enter lower bound for n: '); 
R = input('Enter upper bound for n: '); 
fprintf('
 n \tInner Area \tOuter 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', n, innerA, outerA); 
```
Example 2: Indefinite iteration

What if you want to specify any \( n \), in any order? You promise to use the entry \( \texttt{9999} \) to indicate the end of data entry. Complete the following program to calculate the \( n \)-gon areas. Again use only scalar variables.

\[
\text{n} = \text{input}('\text{Enter integer n: }'); \quad \% \text{ no. of sides of n-gon}
\]

\[
% \text{Compute and display areas of n-gons}
\text{innerA} = (n/2) * \sin(2*\text{pi}/n);
\text{outerA} = n * \tan(\text{pi}/n);
\text{fprintf('n n\text{Inner Area}\text{Outer Area}\n')};
\text{fprintf('d\text{.6f}\text{.6f}\n', n, innerA, outerA)};
\]

Example 3: Initializing variables

Do you need to initialize variables? Consider variable \texttt{total} in the following program:

\[
% \text{Average 10 numbers from user input}
\text{n} = 10; \quad \% \text{number of data values}
\text{total} = 0; \quad \% \text{current sum (initialized to zero)}
\text{k} = 1; \quad \% \text{initialize counter}
\text{while} \ (k<n)
\quad \% \text{read and process input value}
\quad \text{num} = \text{input}('\text{Enter a number: }');
\quad \text{total} = \text{total} + \text{num};
\quad \% \text{update counter}
\quad \text{k} = \text{k} + 1;
\end{while}
\text{ave} = \text{total}/\text{n} \quad \% \text{average of n numbers}
\]

Brute-force algorithm to find minimum function value

- Start at \( x = L \)
- Set a current minimum function value—"minimum found so far"
- Until the end point \((x = R)\) is reached, repeat the following:
  - Calculate \( f(x) \)
  - Compare \( f(x) \) with the minimum found so far and update if necessary
  - Increment value of \( x \)
The while loop

while (expression)

    statements to execute if
    expression evaluates to true

dev
Pattern for doing something \( n \) times

\[
k= 1;
\]

\textbf{while} \( k \leq n \)

\[
\text{\% do something}
\]

\[
\text{\% \ldots }
\]

\[
k= k + 1;
\]

\textbf{end}
Pattern for doing something
an indefinite number of times

% initialization

% ...

while not stopping signal

% do something

% ...

% update status (variables)

% ...

end