Previous Lecture:
- Iteration using \texttt{while}

Today’s Lecture:
- Nested loops
- Developing algorithms

Announcements:
- Discussion this week in computer labs. \textit{Read Insight §3.2} before discussion section.
- \textbf{Project 2} due Thursday at 11pm
- We do not use \texttt{break} in this course
- Make use of Piazza, office hrs, and consulting hrs
Common loop patterns

Do something \( n \) times

\[
\text{for } k = 1:1:n \\
\quad \text{% Do something} \\
\text{end}
\]

Do something an indefinite number of times

\[
\text{while } ( \text{not stopping signal} ) \\
\quad \text{% Initialize loop variables} \\
\quad \text{% Do something} \\
\quad \text{% Update loop variables} \\
\text{end}
\]
for-loop or while-loop: that is the question

- **for-loop**: loop body repeats a *fixed* (predetermined) number of times.

- **while-loop**: loop body repeats an *indefinite* number of times under the control of the “loop guard.”
What is the last line of output?

```matlab
x = 1;
disp(x)
y = x;
while y==x && x<=4 && y<=4
    x = 2*x;
disp(x)
end
```

A: 1  B: 2  C: 4  D: 8
What will be displayed when you run the following script?

```matlab
for k = 4:6
    disp(k)
k = 9;
    disp(k)
end
```

A  
B  
C  

or  

4  
9  
4  
4  

Something else …
Example: Nested Stars
Example: Nested Stars

Draw a black square
- Bigger than the biggest star (at least 2 times radius of star)
- Center at (0,0)

Draw a sequence of stars
- Stars alternate in color
- Stars get smaller
  - radius $r=1$ to start
- $1^{st}$ star smaller than the sqr
- When to stop?
  - when $r$ is small

nestedStars.m
Knowing how to draw

How difficult is it to draw
Pattern for doing something $n$ times

\[
n = \_\_\_\_
\]
\[
\textbf{for} \ k = 1 : n
\]
\[
\% \ \text{code to do}
\]
\[
\% \ \text{that something}
\]
\[
\textbf{end}
\]
x = 0; y = 0; % figure centered at (0,0)

s = 2.1; % side length of square
DrawRect(x-s/2,y-s/2,s,s,'k')

r = 1; k = 1;
while r > 0.1 % r still big
    % draw a star
    if rem(k,2) == 1 % odd number
        DrawStar(x,y,r,'m') % magenta
    else
        DrawStar(x,y,r,'y') % yellow
    end
    % reduce r
    r = r/1.2;
    k = k + 1;
end
for c = 0:2:8

x = c; y = c; % figure centered at (c,c)

s = 2.1; % side length of square
DrawRect(x-s/2,y-s/2,s,s,'k')

r = 1; k = 1;
while r > 0.1 %r still big
    % draw a star
    if rem(k,2) == 1 %odd number
        DrawStar(x,y,r,'m') %magenta
    else
        DrawStar(x,y,r,'y') %yellow
    end
    % reduce r
    r = r/1.2;
    k = k + 1;
end
end
Pattern for doing something $n$ times

\[ n = \_\_\_\_ \]
\[ \textbf{for} \quad k = 1:n \]
\[ \% \text{ code to do} \]
\[ \% \text{ that something} \]
\[ \textbf{end} \]
Example: Are they prime?

- Given integers \( a \) and \( b \), write a program that lists all the prime numbers in the range \([a, b]\).
- Assume \( a > 1 \), \( b > 1 \) and \( a < b \).
Example:  Are they prime?
Subproblem:  Is it prime?

- Given integers $a$ and $b$, write a program that lists all the prime numbers in the range $[a, b]$.
- Assume $a>1$, $b>1$ and $a<b$.
- Write a program fragment to determine whether a given integer $n$ is prime, $n>1$.
- Reminder:  $\text{rem}(x,y)$ returns the remainder of $x$ divided by $y$. 
Example: Are they prime?
Subproblem: Is it prime?

- Given integers \( a \) and \( b \), write a program that lists all the prime numbers in the range \([a, b]\).
- Assume \( a > 1 \), \( b > 1 \) and \( a < b \).
- Write a program fragment to determine whether a given integer \( n \) is prime, \( n > 1 \).
- Reminder: \( \text{rem}(x, y) \) returns the remainder of \( x \) divided by \( y \).
Start:
  \( \text{divisor} = 2 \)

Repeat:
  \( \text{rem}(n, \text{divisor}) \)
  \( \text{divisor} = \text{divisor} + 1 \)

End:
  \( \text{rem}(n, \text{divisor}) = 0 \)
  \( \text{divisor} \leq n \) ?

\[ \text{divisor} = 2; \]
\[ \text{while } (\text{rem}(n, \text{divisor}) \neq 0) \]
\[ \text{divisor} = \text{divisor} + 1; \]
\[ \text{end} \]
\[ \text{if } (\text{divisor} = n) \]
\[ \text{disp}('prime') \]
\[ \text{else} \]
\[ \text{disp}('composite') \]
\[ \text{end} \]
% Given n, display whether it is prime

divisor = 2;
while ( rem(n, divisor) ~= 0 )
    divisor = divisor + 1;
end
if (divisor == n)
    fprintf(‘%d is prime\n’, n)
else
    fprintf(‘%d is composite\n’, n)
end
for n = a:b

%Given n, display whether it is prime
divisor = 2;
while ( rem(n,divisor)~=0 )
    divisor = divisor + 1;
end
if (divisor==n)
    fprintf('%d is prime\n', n)
else
    fprintf('%d is composite\n', n)
end

end
Example: Times Table

Write a script to print a times table for a specified range.

<table>
<thead>
<tr>
<th>Row headings</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
</tr>
</tbody>
</table>
Developing the algorithm for the times table

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
</tr>
</tbody>
</table>
Developing the algorithm for the times table

- Look for patterns
  - Each entry is \( \text{row}\# \times \text{col}\# \)
  - Row\#, col\# increase regularly
- \( \Rightarrow \) Loop!!!
- What kind of loop?
  - for-loop—since the range of the headings is specified and the increment is regular
  - for each row\#, get the products with all the col\#s. Then go to next row\# and get products with all col\#s, …
- \( \Rightarrow \) Nested loops!
- Details: what will be the print format? Don’t forget to start new lines. Also need initial input to specify the range.

<table>
<thead>
<tr>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
</tr>
</tbody>
</table>
disp('Show the times table for specified range')
lo= input('What is the lower bound? ');
hi= input('What is the upper bound? ');

mTable.m
Rational approximation of $\pi$

- $\pi = 3.141592653589793\ldots$
- Can be closely approximated by fractions, e.g., $\pi \approx 22/7$
- Rational number: a quotient of two integers
- Approximate $\pi$ as $p/q$ where $p$ and $q$ are positive integers $\leq M$
- Start with a straightforward solution:
  - Get $M$ from user
  - Calculate quotient $p/q$ for all combinations of $p$ and $q$
  - Pick best quotient $\rightarrow$ smallest error