



More on Loops & More on Developing Algorithms

Lecture 6 (Feb 9)
CS100M - Spring 2006

Announcements

- Prelim 1
 - 2/23 at 7:30pm
 - Contact Kelly Patwell (see website) if you have an unresolved university-scheduled conflict
- Sign out a *clicker* from the Engineering Library
 - Register it online:
<http://instruct1.cit.cornell.edu:8000/clickers/cs100m.php>
 - You will use your clicker to answer in-class questions and to take in-class quizzes
- For next week, section will be back in the lab

Topics

- Reading: CFile Chapter 3
- Recall
 - Matlab iteration
 - For-loop
 - While-loop
- Plans for today
 - More on iteration
 - For-loop vs. while-loop
 - More on developing algorithms

For-Loops & While-Loops

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

```
while <boolean condition>
    Statements to execute (also called loop body)
end
```

- How do we know which one to use?
 - For-loop: Loop body is repeated a fixed, predetermined number of times
 - While-loop: Loop body is repeated an indefinite number of times under the control of a boolean condition

We Don't Have to Use For-Loops At All

- This for-loop


```
for i = 3:2:39
    % do something
end
```
- Can be replaced by a while-loop without significantly changing what the program does


```
i = 3;
while i <= 39
    % do something
    i = i + 2
end
```
- The loop behaviors aren't quite identical
 - The values left in variable *i* after the loop-end aren't the same

Why Both For-Loops and While-Loops?

- We could do without the for-loop
 - Because any for-loop can be replaced with a while-loop
- But the for-loop kind of loop occurs so often, it's useful to have a compact, easy-to-read way to build a for-loop
 - It takes more typing to create an equivalent while-loop
 - It's easier to read and understand a for-loop than the equivalent while-loop
- Well then, can we get rid while-loops?
 - No, a typical while-loop *cannot* be replaced with a for-loop

Typical Patterns for Loops

- To do something n times
- To do something an indefinite number of times

```

for k = 1:n
    % Do something
    % ...
end

% Initialize loop variables
% ...
while <not stop condition>
    % Do something
    % ...
    % Update loop variables
    % ...
end
    
```

Example: Max

- Goal: Find the max of a sequence of positive numbers
- For-loop or while-loop?
 - While-loop because the length of the sequence is not initially known
- Assume that a non-positive number indicates the end of the sequence
- Algorithm
 - Initialize
 - Get input number
 - while number > 0
 - Compare number to *best*
 - Update *best*, if necessary
 - Get next input number
 - Report *best*
- Basic idea
 - We keep track of the best-value-seen-so-far
 - Each new value is compared to this best value and the best value is updated if necessary

Example: Mode

- Find the *mode* of a non-decreasing, non-negative sequence
- Notes
 - Rules for input aren't specified
 - We should do something reasonable
 - Assume: User provides numbers one at a time
 - Assume: Negative number indicates *Done*
 - We should do something reasonable for illegal input
 - For-loop or while-loop?
- The *mode* is the number that occurs most often
- Examples
 - 1 5 5 6 6 10
 - 2 4 6 8 10 12 14
 - 7 0 5 2 4 6 2 7 0 7

Algorithm for Mode Problem

- Ideas
 - We keep track of the value and how many times it occurred for our "best mode so far"
 - When we find a new number we start a counter for it
 - Every time we increment the counter, we check to see if we have found a better "best mode so far"
- Initialize
 - Get initial number
 - while number >= 0
 - if same as previous
 - Increment counter
 - Compare to best mode & update, if necessary
 - elseif increased (new value)
 - Initialize counter
 - else
 - Bad input: error message
 - Get next number
 - Report best mode seen

Example: Times Table

- Goal: Create a program to print a times table for a specified range
- Assume that the specified range uses positive numbers < 100
- Example: The table for range [2..5] is

2	3	4	5	
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25
- We need *nested* loops
 - A loop to do the rows
 - An inner loop to do each value in a row
- For-loops or while-loops?
 - For-loops because we *know* how often each loop is executed
- How much space should we allow for each number?

Algorithm for Times Table

- Initial version
 - Get the range [low..high]
 - Print the column headings
 - for each row
 - Print the row indicator
 - Print the rest of the row
- Expanded version
 - Get the range [low..high]
 - Make space for row indicator for each value
 - Print the value
 - Make an end-of-line mark for each row
 - Print the row indicator for each value
 - Print the product
 - Make an end-of-line mark

		column headings			
		2	3	4	5
row indicators	2	4	6	8	10
	3	6	9	12	15
	4	8	12	16	20
	5	10	15	20	25

The Savvy Programmer...

- Learns useful *programming patterns* and uses them where appropriate
- Seeks inspiration by working through test data "by hand"
 - Asks, "What am I doing?" at each step
- Decomposes problem into manageable subtasks
 - Refines the problem iteratively, solving simpler subproblems first
- Declares variables for each piece of information maintained when working by hand
 - Writes comments for each such (important) variable
- Remembers to check the problem's *boundary conditions*
- Validates the program by trying it on test data