Branching (if-else)

Lecture 3 (Jan 30)
CS100M – Spring 2007

Announcements

- Project 1
  - Due Thursday, Feb 1 (6pm)
- Section this week
  - Go to your lab, not your classroom
    - Check the course website if you haven't found your lab yet
- Two new Sections have been added
    - Register before attending one of these
  - Section 20 (W at 12:20)
  - Section 21 (W at 1:25)

Topics

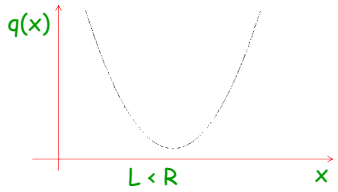
- Recall previous lecture
  - Variables and the assignment statement
  - Input and output
    - Input: `y = input('Give me a y-value.');`
    - Output: `fprintf('The answer is %f.', answer)` `disp('Hi there!')`
- Plans for today
  - Branching; nested branching

Finding the Minimum

Consider the quadratic function

$$q(x) = x^2 + bx + c$$

on the interval [L , R]



Finding the Minimum

Consider the quadratic function

$$q(x) = x^2 + bx + c$$

on the interval [L , R]:

- Which is smaller, q(L) or q(R) ?
- Does q'(x) have a zero in [L , R]?
- What is the minimum value of q(x) in [L , R]?

Min-Finding Algorithm

```
Calculate q(L)
Calculate q(R)
If q(L) < q(R)
  print "q(L) less than q(R)"
Otherwise
  print "q(L) greater than or equal to q(R)"
```

Equivalent Matlab Fragment

```
% Min-Finding Fragment
qL= L*L + b*L + c;          % q(L)
qR= R*R + b*R + c;          % q(R)
if (qL < qR)
  fprintf('qL less than qR\n');
else
  fprintf('qL >= qR\n');
end
```

Relational Operators

<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Do these two code fragments do the same thing?

```
% given x, y
if (x > y)
  disp('alpha')
else
  disp('beta')
end
```

```
% given x, y
if (y > x)
  disp('beta')
else
  disp('alpha')
end
```

A: yes B: no

Do these two code fragments do the same thing?

```
% given x, y
if (x > y)
  disp('alpha')
else
  disp('beta')
end
```

```
% given x, y
if (x > y)
  disp('alpha')
end
if (y >= x)
  disp('beta')
end
```

A: yes B: no

Nested Branching

Goal

- Create a Matlab program to determine the minimum value of

  $q(x) = x^2 + bx + c$

  in the interval [L, R]

- We know how to do this using Calculus
  - The answer has to be one of $q(x_c)$, q(L), or q(R) where x_c is the critical point (where the derivative is zero)
  - But we use $q(x_c)$ only if x_c is in [L, R]

Algorithm Outline

- Compute x_c
- If $x_c \in [L, R]$
  - Answer is $q(x_c)$
- Otherwise
  - Answer is min of q(L) and q(R)

We already know how to do this

Algorithm (with More Detail)

- Compute x_c
- If $L \le x_c \le R$
  - Answer is $q(x_c)$
- Otherwise
  - Compute qL = q(L); compute qR = q(R)
  - If qL < qR
    - Answer is qL
  - Otherwise
    - Answer is qR

We have an if-construct inside another if-construct

Program Fragment

```
% Determine min value of q(x) = x^2 + b*x + c
% in the interval [L, R]
xc = - b/2;                 % Compute xc
if (L <= xc && xc <= R)
    minValue = xc^2 + b*xc + c;
else                        % Compute min of q(L) and q(R)
    qL = L^2 + b*L + c;
    qR = R^2 + b*R + c;
    if (qL < qR)
        minValue = qL;
    else
        minValue = qR;
    end
end
fprintf('Min value is %f\n', minValue)
```

Things to Note

- An if-construct can appear within a branch, just like any other kind of statement
- Matlab (and most other programming languages) treat comparison operators as binary operators
  - Thus some kinds of standard math notation do not work in a Matlab program
    - Math: If 1 < x < 10 then…
    - Matlab: if (1 < x && x < 10)…

Indentation

- Indentation helps make the program readable
- But Matlab doesn't enforce indentation rules
  - Your projects are graded on both correctness and style
    - Appropriate indentation is necessary to achieve a good style grade
  - The Matlab Editor helps with the indentation
    - You can override this, but you shouldn't

Logical And

- How do we check if x_c is in [L, R]?
  - We check $L \le x_c$ and $x_c \le R$
  - In our code: `(L <= xc && xc <= R)`
- Rules for logical and:

x	y	x and y
F	F	F
F	T	F
T	F	F
T	T	T

Logical Or

- Alternately, we could check if x_c is outside of [L, R]
  - We check $x_c \le L$ or $R \le x_c$
  - In our code: `(xc <= L || R <= xc)`
- Rules for logical or:

x	y	x or y
F	F	F
F	T	T
T	F	T
T	T	T

Logical Operators

- Logical and: &&
- Logical or: ||
- Logical not: ~
- Matlab uses 0 for false and nonzero for true
  - Uses 1 for true when Matlab generates it, but will take any nonzero as true in a logical expression
  - Matlab also has predefined logical constants:
    - false (= 0) and true (= 1)

Comparison Operators

- Equal ==
- Not equal ~=
- Less than <
- Greater than >
- Less than or equal <=
- Greater than or equal >=
- Each of these operators produces a boolean result (i.e., the result is either true or false)
- Note use of == to compare for equality

Creating a Program

- As an example, we'll create a program to find areas for some simple shapes
  - Program outline
    - Ask for a shape choice
    - Ask for measurements
    - Report the area
  - We'll use triangle, rectangle, and square

Several Forms of if-Statement

- Short

```
if condition
    statements
end
```

- Medium

```
if condition
    statements
else
    statements
end
```

- Long

```
if condition
    statements
elseif condition
    statements
elseif condition
    statements
else
    statements
end
```

Playing with Comparisons

- Suppose x has the value 5
  - What is the result of typing
    `x < 10`
    in the Matlab Command Window?

    A: false
  - What is the result of typing
    `6 < x`
    in the Matlab Command Window?

    B: true
  - What is the result of typing
    `6 < x < 10`
    in the Matlab Command Window?