Lecture 3: Conditionals

- Previous lecture:
  - Variables & assignment
  - Built-in functions; input & output
  - Programming style (comments, meaningful variable names)

- Today:
  - Writing a program—systematic problem solving
  - Branching (conditional statements)

- Announcements:
  - First project will be posted after lecture; due Feb 23
  - Take advantage of Consulting Hours
  - Take advantage of Ed Discussions
  - Consider enrolling in AEW
  - 1 open seat in LEC 001, DIS 202
Quick review

▪ Variable
  ◦ A named memory space to store a value

▪ Assignment operator: =
  ◦ Let x be a variable that has a value. To give variable y the same value as x, which statement below should you write?

    \[
    x = y \quad \text{or} \quad y = x
    \]

▪ Script (program)
  ◦ A sequence of statements saved in an m-file

▪ ; (semi-colon)
  ◦ Suppresses printing of the result of assignment statement
Tips for writing a program

- Check that you know what is given (or is input, or is assumed)

- Be *goal-oriented*: start by writing the last statement(s) for the program output
  - What is the program supposed to produce? *You know this from the problem statement*
  - Allows you to work backwards from the results

- Name as a variable what you don’t know
  - Helps you break down the steps
  - Allows you to temporarily skip over any part that you don’t know yet how to do
% Compute surface area increase of a sphere in miles^2 given an increase in the radius in inches

r = input('Enter radius r in miles: ');
delta = input('Enter delta r in inches: ');

newr = r + (delta/12)/5280;  % mi

A = 4*pi*r^2;               % mi^2
newA = 4*pi*newr^2;        % mi^2

deltaA = newA - A;         % mi^2

fprintf('Increase in mile^2 is %f.
', deltaA)
Beyond batching

- So far, *all* the statements in our scripts are executed in order.
- We do not have a way to specify that some statements should be executed only under some condition:
  - Want to be able to make decisions.
- We need a new language construct...
Motivation

Consider the quadratic function

\[ q(x) = x^2 + bx + c \]

on the interval \([L, R]\):

- Is the function strictly increasing in \([L, R]\)?
- Which is smaller, \(q(L)\) or \(q(R)\)?
- What is the minimum value of \(q(x)\) in \([L, R]\)?
Problem 1

Write a code fragment that prints “Increasing” if $q(x)$ strictly increases across the interval and “Not increasing” if it does not.
% Quadratic q(x) = x^2 + bx + c
b = input('Enter b: ');
c = input('Enter c: ');
L = input('Enter L: ');
R = input('Enter R, R>L: ');

% Determine whether q increases
% across [L,R]
What are the critical points?

- End points: $x = L$, $x = R$
- $\{ x \mid q'(x) = 0 \}$

\[
q(x) = x^2 + bx + c
\]
\[
q'(x) = 2x + b
\]
\[
q'(x) = 0 \implies x_c = \frac{-b}{2}
\]
The situation

\[ q(x) = x^2 + bx + c \]

\[ x_c = -\frac{b}{2} \]
Does $q(x)$ increase across $[L,R]$?

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

$x_c = -\frac{b}{2}$

No!
So what is the requirement?

% Determine whether q increases % across [L,R] 
xc = -b/2;

if ________________
    fprintf('Increasing\n')
else  % otherwise
    fprintf('Not increasing\n')
end
So what is the requirement?

% Determine whether q increases % across [L,R]
x_c = -b/2;

if ________________
    fprintf('Increasing\n')
else
    fprintf('Not increasing\n')
end
% Determine whether q increases
% across [L,R]
xc = -b/2;

if xc <= L
    fprintf('Increasing\n')
else
    fprintf('Not increasing\n')
end
Problem 2

Write a code fragment that prints

“qleft is smaller”
if q(L) is smaller than q(R).
If q(R) is smaller print
“qright is smaller.”
Algorithm v0

Calculate $q(L)$
Calculate $q(R)$

If $q(L) < q(R)$
    print “$q_{left}$ is smaller”

Otherwise
    print “$q_{right}$ is smaller”
Algorithm v1.0

Calculate $x_c$

If distance $x_cL$ is smaller than distance $x_cR$
   
   print “qleft is smaller”

Otherwise

   print “qright is smaller”
Do these two 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
Algorithm v1.1

Calculate $x_c$

If distance $x_c L$ is smaller than distance $x_c R$
   print “qleft is smaller”

Otherwise
   print “qright is smaller or equals qleft”
Algorithm v1.2

Calculate $x_c$

If distance $x_cL$ is same as distance $x_cR$
  print “$q_{left}$ and $q_{right}$ are equal”

Otherwise, if $x_cL$ is shorter than $x_cR$
  print “$q_{left}$ is smaller”

Otherwise
  print “$q_{right}$ is smaller”
% Which is smaller, q(L) or q(R)?

xc = -b/2;  % x at minimum
if (abs(xc-L) == abs(xc-R))
    disp('qleft and qright are equal')
elseif (abs(xc-L) < abs(xc-R))
    disp('qleft is smaller')
else
    disp('qright is smaller')
end
Algorithm v0.2

Calculate $q(L)$
Calculate $q(R)$

If $q(L)$ equals $q(R)$
    print "$q_{left}$ and $q_{right}$ are equal"

Otherwise, if $q(L) < q(R)$
    print "$q_{left}$ is smaller"

Otherwise
    print "$q_{right}$ is smaller"
% Which is smaller, q(L) or q(R)?

qL = L*L + b*L + c;  % q(L)
qR = R*R + b*R + c;  % q(R)
if (qL == qR)
    disp('qleft and qright are equal')
elseif (qL < qR)
    disp('qleft is smaller')
else
    disp('qright is smaller')
end
% Which is smaller, q(L) or q(R)?

qL = L*L + b*L + c;  % q(L)
qR = R*R + b*R + c;  % q(R)
if (qL == qR)
    disp('qleft and qright are equal')
    fprintf('q value is %f\n', qL)
elseif (qL < qR)
    disp('qleft is smaller')
else
    disp('qright is smaller')
end
Consider the quadratic function

\[ q(x) = x^2 + bx + c \]

on the interval \([L, R]\):

What if you only want to know if \(q(L)\) is close to \(q(R)\)?
% Is $q(L)$ close to $q(R)$?

tol = 1e-4;  % tolerance
qL = L*L + b*L + c
qR = R*R + b*R + c
if (abs(qL-qR) < tol)
    disp('qleft and qright similar')
end

else is optional in an if-statement. This if-statement without else is correct.
The **if** construct

```markdown
if boolean expression1
    statements to execute if expression1 is true
elseif boolean expression2
    statements to execute if expression1 is false but expression2 is true
else
    statements to execute if all previous conditions are false
end
```

Can have any number of **elseif** branches but at most one **else** branch
Things to know about the if construct

▪ **At most one** branch of statements is executed
▪ There can be **any number of elseif** clauses
▪ There can be **at most one else** clause
▪ The **else** clause **must be the last clause** in the construct
▪ The **else** clause **does not have a condition** (boolean expression)
Problem 3

Write a code fragment that prints “Inside” if $xc$ is in the interval and “Outside” if it is not.
Is $x_c$ in the interval $[L,R]$?

$\begin{align*}
q(x) &= x^2 + bx + c \\
x_c &= -\frac{b}{2}
\end{align*}$

No!
Is $x_c$ in the interval $[L,R]$?

- What mathematical condition tells us the answer? (in English)

$$x_C \geq L \text{ AND } x_C \leq R$$
Logical operators

`&&` logical **and**: Are both conditions true?
E.g., we ask “is \( L \leq x_c \) and \( x_c \leq R \)?”
In our code: \( L \leq x_c \) **and** \( x_c \leq R \)

`||` logical **or**: Is at least one condition true?
E.g., we can ask if \( x_c \) is outside of \([L,R]\),
i.e., “is \( x_c \leq L \) or \( R \leq x_c \)?”
In code: \( x_c \leq L \) **or** \( R \leq x_c \)

`~` logical **not**: Negation
E.g., we can ask if \( x_c \) is **not outside** \([L,R]\).
In code: \( \lnot (x_c \leq L \) **or** \( R \leq x_c \) \)