Selection and Iteration

ENGR 151, Lecture 21: 24 Nov 14

Announcements

- Project 7 due Wed 26 Nov
  - Explicit iteration and selection not allowed
  - Don't forget Problem E…

Project Poll

- Which was your favorite C++ project?

A. #2 RPSLK
B. #3 Production Planning
C. #4 Information Retrieval
D. #5 Monopoly
E. #6 Algorithmic Trading

Project Poll

- Which was your *least* favorite C++ project?

A. #2 RPSLK
B. #3 Production Planning
C. #4 Information Retrieval
D. #5 Monopoly
E. #6 Algorithmic Trading

Selection in MATLAB

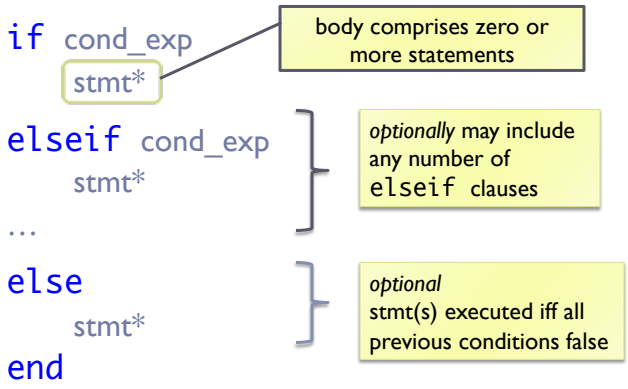
- Two primary selection constructs:
  - `if`
  - `switch`
- Both enable selection of computational steps based on condition expressions

What is a Difference between MATLAB `if` and C++ `if` ?

A. No parentheses required around conditional expression
B. `elseif` construct combining `else` and a nested `if`
C. Required `end` at end of statement
D. All of the above
E. None of the above

Selection using `if`

- General form of the `if` construct in MATLAB:



Example `if` Statement

```
x = input('Enter a number: ');

if x > 0
    disp('The number is positive.\n');
elseif x < 0
    disp('The number is negative.\n');
else
    disp('The number is zero.\n');
end
```

Nested if

- The body of an `if` may itself include `if` statements

```
d = b^2 -4*a*c;
if a ~= 0
  if d < 0
     disp('complex roots');
  else
     x1 = (-b + sqrt(d))/(2*a);
     x2 = (-b - sqrt(d))/(2*a);
  end
end
```

Selection on Logic Vectors

```
x = [10 20 30 40];
if x > 20
  disp('true!')
elseif x < 20
  disp('false!')
else
  disp('tralse!')
end
```

What does this print?

A. true!
B. false!
C. tralse!
D. something else
E. nothing

Example: Vector Equality

- suppose x and y are vectors of the same length

```
if x == y
  disp('vectors are equal')
else
  disp('vectors not equal')
```

works as intended

```
if not(x == y)
  disp('vectors not equal')
else
  disp('vectors are equal')
```

does *not* work as intended (why not?)

```
if not(all(x == y))
  disp('vectors not equal')
else
  disp('vectors are equal')
```

works as intended

Switch

```
switch expr
  case case_expr
      stmt*
  case case_expr
      stmt*
  …
  otherwise
      stmt*
  end
```

Scalar or string expression

Execute stmt(s) after case with label corresponding to switch expr

body for each case comprises zero or more statements

optional
stmt(s) executed iff all previous conditions false

Switch: Cases

```
switch expr
  case {expr1,...}
      stmt*
  case case_expr
      stmt*
  ...
  otherwise
      stmt*
  end
```

set of expressions, in curly braces

Multiple labels for single case allowed

Switch Example

```
switch (day)
    case {1, 7}
          disp('Weekend');
    case 2
          disp('Monday');
    case 3
          disp('Tuesday');
    case 4
          disp('Wednesday');
    case 5
          disp('Thursday');
    case 6
          disp('Friday');
    otherwise
          disp('out of range');
    end
```

Iteration in MATLAB

- iteration constructs:
  - `while`
  - `for`
- Elt-by-elt operations on arrays can also effectively accomplish iterative tasks without explicit iteration programming constructs

While in MATLAB

```
while cond_expr
      stmt*
end
```

While Loop Example

```
sum = 0;
x = input('Enter a number: ');
while x >= 0
   sum = sum + x;
   x = input('Enter a number: ');
end
display(sum);
```

MATLAB for Statement

- Which is a syntactically valid `for` header in MATLAB?

A. `for (idx = 0; idx < N; idx = idx + 1)`
B. `for idx = 0, idx < N, idx = idx + 1`
C. `for idx = 0 to N`
D. `for idx = 0:N`
E. None of the above

For in MATLAB

```
for counter = vector_expr
    stmt*
end
```

Execute the loop body once for each elt of the vector, with counter bound to that elt value

For Example

```
val = 1;
for n = 1:10
  val = val * n;
end
```

val = 10!

```
val = 1;
for n = 1:2:10
  val = val * n;
end
```

val = 1 × 3 × 5 × 7 × 9

For Another Example

```
val = 1;
for n = [2 10 -3 6]
  val = val * n;
end
```

val = 2 × 10 × –3 × 6

```
val = ...
 prod([2 10 -3 6]);
```

val = 2 × 10 × –3 × 6

For with Vector Counters

```
for  counter = array_expr
       stmt*
end
```

- The counter can also be assigned to an array with more than one dimension
- In this case the counter takes on the value of an entire column one column at a time

Exercise

```
a = [1:3; 2:2:6];
x = ones(2,1);
for b = a
   x = x.*b;
end
```

What is x after executing this code?

A [3 48]

B [3 12]

C [6 48]

D [6 12]

E None of these

Evaluating Counting Loops

```
x = [0:2];
for i=1:length(x)
   x = [x i];
end
```

x = [0 1 2 1 2 3]

```
x = [0:2]; i=1;
while i<=length(x)
   x = [x i];
   i = i + 1;
end
```

infinite loop

- The for statement vector expression is evaluated *once*
- The while statement condition is evaluated *each iteration*

break and continue

- Alter the normal flow of control in a loop (for or while)
- break terminates the loop
- continue terminates the loop body: current pass only

break and continue Example

```
for i = 1:2:11
  if mod(i, 3)==0
     continue;
  end
  disp( i );
end
```

```
1
5
7
11
```

```
for i = 1:2:11
  if mod(i, 3)==0
     break;
  end
  disp( i );
end
```

```
1
```

Iteration and Vectorization

- Explicit iteration constructs:
  - `while`
  - `for`
- Elt-by-elt operations on arrays can also effectively accomplish iterative tasks without explicit iteration programming constructs
- Vectorizing calculation in this way is generally
  - more efficient
  - more natural (once you get used to it)

than the corresponding for or while construct

Exercise: Which is an Equivalent Vectorized Expression?

```
a = zeros(1,100);
for i = 1:100
  a(i) = exp(-0.2 * i);
end
```

A
```
a = exp( -0.2 * [1 100] );
```

B
```
a = exp( -0.2 * [1:100] );
```

C
```
i = 1:100;
a = exp( -0.2 * i );
```

D
```
i = 1:100;
a(i) = exp( -0.2 * i );
```

E
```
i = 1:100;
a(i) = exp( -0.2 .* i );
```

Vectorize This

```
g = rand(1,10);
for i = 1:2:length(g)-1
  g(i) = g(i+1);
end
```

`rand` fills a matrix with random numbers

```
g = rand(1,10);
g(1:2:9) = g(2:2:10);
```

Vectorize This

```
g = rand(1,11);
g(1:2:10) = g(2:2:11);
```

What is the result if we replace the assignment above with:

```
g(1:2:end) = g(2:2:end)
```

A. same as above
B. same, except `g(11)` gets assigned 0
C. same, except `g(11)` gets assigned arbitrary value
D. error
E. none of the above

Exercise: Vectorize This

```
for a = 1:size(M,1)
  for b = 1:size(M,2)
    if( mod(a,2)==0 and mod(b,3)==0 )
      M(a,b) = M(a-1,b-1);
    end
  end
end
```

A `M(1:2:end,1:3:end) = M(1:1:end,1:2:end);`

B `M(2:2:end,2:3:end) = M(1:2:end,1:3:end);`

C `M(2:2:end,3:3:end) = M(1:2:end,1:3:end);`

D `M(2:2:end,3:3:end) = M(1:2:end,2:3:end);`

E None of the above

.p Files

- MATLAB stores function definitions in an internal format called pcode
  - avoids repeating some interpreter processing (e.g., parsing text) every time function is invoked
  - a form of compilation, but not to native machine language
- Can save pcode to a dedicated file with command:
  - `pcode` fn_name
  - creates file fn_name`.p`
  - protected binary format, not readable in text editor