W1005
Intro to CS and Programming in MATLAB

Control Flow

Fall 2014

Instructor: Ilia Vovsha

http://www.cs.columbia.edu/~vovsha/w1005

Outline

- Array Manipulation
- Control Flow

Array Manipulation

- Special functions for arrays:
  - sum(), prod(), mean(), max(), min()
  - First argument we pass to each function is the array
  - What if array is a matrix, what does sum(X) mean?
- Useful routines:
  - sort(), find(), sortrows(), any()
  - Same issue as above, what does sort a matrix mean?

Array Manipulation

- Special functions for arrays:
  - sum(), prod(), mean(), max(), min()
- Suppose M is a matrix, then:
  - TScol = sum(M); TScol is a row vector, each element is the column sum
  - TSrow = sum(M,2); TSrow is a column vector, each element is the row sum
  - TS = sum(sum(M)); TS is the total sum of all elements in the matrix
  - Can also obtain the location of the max/min value (if argument is a vector):

    [val, loc] = min(M);

Array Manipulation

- Useful routines:
  - sort(), find(), sortrows(), any()
- Function sort() sorts elements in *ascending* (default) order. Can specify the order explicitly:
  - sort_list = sort(M, 'descend');
  - If M is a matrix, sorts each column independently
  - As above, can sort across columns: sort(M,2, 'ascend')
  - If consistency between rows is required, use sortrows()

Array Manipulation

- Function find() returns indices of elements which satisfy some condition:
  - find(M) Return indices of nonzero elements of M (default)
  - find(M == 1) Return indices of elements of M equal to one
  - find(M >= 2 & M < 6) Return indices that satisfy both conditions
- If M is a matrix, the returned index is a single value which is a bit annoying. Solution: use ind2sub(), sub2ind() to convert between indices
- [r,c] = ind2sub(size(M),index)

Control Flow

- Without flow, the scripts have no life
- In general: a control flow statement begins with some keyword and ends with the keyword end
- General Form (if):

```
if (conditional statement)
        body
end
```

- Example:

```
if  x == 1
        y = 5;
end
```

Control Flow – Cond. Statement

- General Form (if):

```
if (conditional statement)
        body
end
```

- Conditional statements can be with or without ()
- Typically use logical and relational operators in the condition:
  - | (OR), & (and), ~ (NOT)
- Good idea to use () when the statement is a long one
- Alternative: define a variable in the previous line
  - T = (x == 2 || y == 3) & x > 5;
  - if (T) …

Control Flow (if – else)

- General Form (if - else):

```
if (conditional statement)
        body1
else
        body2
end
```

- Example:

```
if  x == 1
        y  =  5;
else
        y  =  2;
end
```

Control Flow (if – elseif)

- General Form (if – elseif):

```
if  (cond. statement1)
        body1
elseif  (cond. statement2)
        body2
else
        body3
end
```

- Example:

```
if  x == 1
        y  =  5;
elseif   x  == 2
        y  =  3;
else
        y  =  2;
end
```

Control Flow – Loops (for)

- Used for repetition
- General Form (for):

```
for “iteration variable”
        body
end
```

- Notice how the iteration variable is iterated over a row vector
- Example:

```
for  i  = 1:5
        disp(i+1);
end
```

Control Flow – Loops (while)

- General Form (while):

```
while (cond. statement)
        body
end
```

- Example:

```
x = 0;
while  x < 5
        x  = x + 1;
        disp(x);
end
```

Control Flow – Loops (break)

- Example (continue):

```
y = 0;
for  x  =  1:10
      if  (x == 3)
            continue;
      end
      y = y + 1;
end
```

- What is y = ?

- Example (break):

```
y = 0;
for  x  =  1:10
      if  (x == 3)
            break;
      end
      y = y + 1;
end
```

- What is y = ?

Control Flow – Caveat

- Loops are extremely inefficient in MATLAB, avoid them like the plague!
- Alternatives?
  - Built-in functions (find, sort, rand)
  - Pre-allocate memory (initialize array before entering loop)

Exercise (In Class)

- Control flow & array manipulation
- Write a script that takes two (hard-coded) vectors {A,B}. For every element of A, print the element of B at the corresponding index (index = element of A). At the end of the procedure, print the sum and product of the printed elements

Switch Statement – Usage

- When?
  - More than 2-3 choices
  - Choice is based on a common expression
- How?

```
switch  switch_expr                    % Expression is a scalar or a string
    case  case_expr                    % Match 'case_expr'
        statement(s)
    case {case_expr1,case_expr2}       % Match any case in the array
        statement(s)
    otherwise
        statement(s)
end
```

Switch Statement – Usage

- Brackets {} are not required unless you wish to execute the same code for more than one case
- ‘break’ statements are redundant. Unlike C++ for example, there is no ‘falling through’
- ‘otherwise’ is optional