

5th Annual (ETCS) Opportunity Banquet

Friday April 4, 2008, Walb Union Ballroom

“Extreme Engineering: The Space Elevator”

Dr. Brad Edwards 7:30-8:30 pm

Career Fair & Networking Dinner

4:30-7:30 pm

Advanced Registration Required for Career Fair & Dinner

\$20.00 IPFW Students

\$30.00 Faculty/Staff

\$35.00 General Public

Ticket sales to only ETCS students, faculty, and staff until March 21

Programming 3: Switch Structure

By the end of this class you should be able to:

- use a switch statement
- create cell arrays & use them in a switch statement
- use the menu command

Text Section 4.6

- handouts: switch handout2
static friction example
- files: see structured programming examples (coming to website soon)
- Conditional Statements/Branching
 - one of the key structures of structured programming
 - if/elseif/else statements
 - the switch structure (switch/case statements)

Review: if statements - general structure

if *logical statement 1*
Statement Group 1 (if 1 true)

elseif *logical statement 2*
Statement Group 2
(if 1 is false & 2 is true)

else
Statement Group 3 (if all are false)

end

variations

- can have multiple elseif s
- do not have to have a final else
- max one else and it must be after all elseif s

Msc reminders

- NaN => not a number
- remember the == in logical comparisons
- Good programming practice: insure the output variable has an assigned value

Practice Exercise

Write a function that accepts a score from 1-100 and then returns a letter grade based on a usual 10% curve (90-100 = A, 80-90=B, 70-90 = C ...). Inputs and outputs should happen on the command line.

Goal		
Input		
Output		
Calc.		

Practice Problem

set in pairs

work on table on next page with class

have work in pairs on program

Goal	Develop a function that will determine letter grade from % score on a standard 10% curve	
Input	Score	%
Output	Grade	Letter
Calc.	89 → B 90 → ? 76 → C 66 → D 55 → F (test all branches)	

Practice Problem

Write a function that accepts a score from 1-100 and then returns a letter grade based on a usual 10% curve (90-100 = A, 80-90=B, 70-90 = C ...). Inputs and outputs should happen on the command line.

Variations

Try using elseifs

Try using nested ifs

Solution function 1

using nested if statements, nested in the "true" portion of each loop.

Notice:

1. The grade is not simply displayed but saved to a variable and then returned on the command line so it can be stored in the main program.
2. Leaving off semicolons does allow the output to be displayed as well

```
function Grade = Curve(score)
% function Grade=Curve(score)
% Determines the letter grade given
% a % score based on a 10%/grade
% Palm problem 4-17
if score >=60
    Grade = 'D'
    if score >= 70
        Grade = 'C'
        if score >= 80
            Grade = 'B'
            if score >=90
                Grade ='A'
            end
        end
    end
end
else
    Grade ='F'
end
```

Solution function 2

using sequential if nested in an outer if statement.

Introductory comments in all of these examples are abbreviated for these notes. Full comments should include the call form, all variables, author and date

```
function Grade = Curve(score)
% function Grade=Curve(score)
% Determines the letter grade given
% a % score based on a 10%/grade
% Palm problem 4-17
if score >=60
    Grade = 'D'
    if score >= 70
        Grade = 'C'
    end
    if score >= 80
        Grade = 'B'
    end
    if score >=90
        Grade ='A'
    end
end
else
    Grade ='F'
end
```

Solution function 3

using nested if in the "false" segment (i.e., after the else statement).

notice: in this example I chose to use semicolons to suppress the echo print.

```
function Grade = Curve(score)
% function Grade=Curve(score)
% Determines the letter grade given
% a % score based on a 10%/grade
% Palm problem 4-17
if score >= 90
    Grade = 'A';
else
    if score >= 80
        Grade = 'B';
    else
        if score >= 70
            Grade = 'C';
        else
            if score >=60
                Grade ='D';
            else
                Grade ='F';
            end
        end
    end
end
end
```

Solution function 4

Using the else if structure throughout

```
function Grade = Curve(score)
% function Grade=Curve(score)
% Determines the letter grade given
% a % score based on a 10%/grade
% Palm problem 4-17
if score >= 90
    Grade = 'A'
elseif score >= 80
    Grade = 'B'
elseif score >= 70
    Grade = 'C'
elseif score >=60
    Grade = 'D'
else
    Grade = 'F'
end
```

Switch Statement (Handout) Equivalent English Statement

switch what you do based on the value of X
 for the case where X equals _____ do ...
 for the case where X equals _____ do ...

```
function angle = compass( direct )
% Lots of Comments
```

Example:

Switch
Compass
Points to
Degrees

```
switch direct
case 'N'
    angle = 0;
case 'W'
    angle=90;
case 'S'
    angle = 180;
case 'E'
    angle = 270;
otherwise
    disp('Error')
    angle =NaN
end
```

General Format

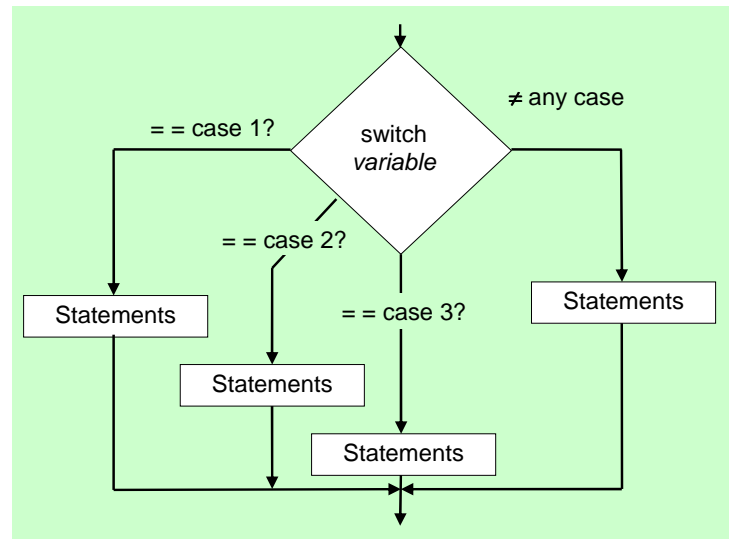
```
switch variable or expression
case value
    statements

case value
    statements

case value
    statements

...
otherwise
    statements

end
```



- issues/description
 - comparison
switch statement *variable == case statement value*
no comparison operator used or allowed.
 - can compare to multiple values using a cell array (later)
- Example with single value: Compass direction
 Input compass direction: N, E, S, W to
 Output compass value: 90, 180, 270, 360 degrees

Related Aside: Cell Arrays Allow multiple conditions in a switch structure

- an array of arrays
- keeps different types/sizes of variables together

Create using:

- curly brackets { } & a list of cells
- num2cell() - to convert a numerical array

Cell Arrays: Review & Follow-the-Leader

How they are created:

enclose set of cells in curly braces {cases}
 cases are delimited by commas
 >> y = {3, 5, 5} notice difference from
 >> y = [3, 5, 5]

use `num2cell()` to convert *number array* to *cell array*
 >> x = [3 4 6]
 >> y = num2cell(x)

What they are: An array of arrays
 e.g. [1,2,3] with [1,2] with 'text'
 >> y = {[1,2,3], [1,2], 'text'}

Why needed/used:

1. Associate related values of different data types
 e.g., a vector of column headers and the associated columns of data
 >> y = {'Height', [8.3, 10.2, 12.3, 15.3, 16.8, 18.0]}
2. Store text strings of different length
 e.g. 'green' 'red' 'blue'
 >> y = {'green'; 'red'; 'blue'}
 (note: y = ['green'; 'red'; 'blue'] results in an error message.)
3. Used in a **case statement** to distinguish between every element in vector must be matched → use a standard array
 any element in the vector can be matches → use a cell array

This latter situation is the point of introducing them here - they are quite useful in switch-case structures.

Examples problems requiring with multiple options

1. Identifying a letter capital or small (see next slide)
 - N or n, S or s, E or e,
2. Can also do the opposite problem: given degrees convert to quadrants.

```
function angle = compass( direct )
% Lots of Comments
```

Switch example with multiple matches:

Switch Compass Points (both cases) to Degrees

```
switch direct
case {'N', 'n'}
    angle = 0;
case {'W', 'w'}
    angle=90;
case {'S', 's'}
    angle = 180;
case {'E', 'e'}
    angle = 270;
otherwise
    disp('Error')
    angle =NaN
end
```

Switch Exercise - Calculating Static Friction

Handout: Program Dev. Worksheet

the force (F) required to start a weight (w) moving is given by:

$$F = \mu w$$

~ static coefficient of frictions (μ)	
Metal on metal	0.20
Wood on wood	0.35
Metal on wood	0.40
Rubber on concrete	0.70

Write a program to calculate:

- the force required to start an object moving
- given the weight of the object and the material combination.
- Use a switch structure

```
function F = start1(W,M)
% lots of comments

% Select the correct coefficient of static friction
switch M
case 1
    mu = 0.2;
case 2
    mu =0.35;
case 3
    mu=0.40;
case 4
    mu =0.7;
otherwise
    disp('ERROR: invalid materials choice')
    mu=NaN;
end

% Calculate the force required to get object moving
F=mu*W;
```

Static Friction Exercise with Command Line inputs

Notes:

- Switch Structure chooses correct mu
- then calculation is done
- inputs are at the command line, matching the Program Development Worksheet
- Variables used match the variables defined in the Program Development Worksheet.

Static Friction with Interactive Inputs (2 pages)

```
function F=start3
```

```
% S.Moor March 2005
```

```
% Calculates the static friction for one of four specific cases
```

```
% given the case and the weight. Input is requested using a button menu
```

```
% function F=start3
```

```
%
```

```
% The four Cases
```

```
% 1. Metal on metal
```

```
% 2. Wood on wood
```

```
% 3. Metal on wood
```

```
% 4. Rubber on concrete
```

```
% variables
```

```
% F = force required to overcome static friction (lbf)
```

```
% W = weight of the object (lbm)
```

```
% M = a number representing the material selections as shown above
```

```
% mu = the coefficient of static friction
```

Menu Statement - creates a push button GUI

```
>> x = menu('choice request', 'option1', 'option2', ...)
```

Menu Statement and Switch Structure are often used together

Try adding menu input of the material to your static friction program.

```
% Request input data
```

```
W=input('What is the weight of the object in lbs.? ');
```

```
M = menu('Choose a Material Combination','Metal on metal',...
```

```
'Wood on wood','Metal on wood','Rubber on concrete');
```

```
% Select the correct coefficient of static friction
```

```
switch M
```

```
case 1
```

```
mu = 0.2;
```

```
case 2
```

```
mu = 0.35;
```

```
case 3
```

```
mu = 0.40;
```

```
case 4
```

```
mu = 0.7;
```

```
otherwise
```

```
disp('ERROR: invalid materials choice')
```

```
mu = NaN;
```

```
end
```

```
% calculate the static friction
```

```
F = mu * W;
```