Week 2 Lab

EG1002 Computing for Engineers

Semester 1, 2013

1 Introduction

1.1 Aim

To become familiar with the structure, operation and constraints of the

- *if* statement
- *switch* statement
- *for* loop
- *while* loop

1.2 Assessment

This lab is assessable and will count towards your final grade, as specified in the subject outline. There are written answers that you must complete as you work through the lab. You may write your answers in a word processing document or on paper. **Your tutor will give you a mark at the end of your lab session.** Do not leave the lab room without getting a mark, or else you will receive zero marks.

2 *if* statement in MATLAB

Terminology

- To instruct a computer to make a simple decision if a condition is true, an *if* statement is used.
- It is the simplest logical check you have at your disposal.
- The syntax of an *if* statement will vary between programming languages, but the general structure is as follows. Note the use of the *elseif* and *end* keywords.

```
if condition1
  code to execute if condition 1 is TRUE
elseif condition2
  code to execute if condition 2 is TRUE
else
  code to execute if no conditions are met
end
```

- A portion of code will execute if the preceding condition is TRUE, otherwise if no conditions are met the code following the *else* statement is executed.
- Conditions use equality(==), inequality (~=) or relational logic (<, <, <=, >=).

1. On LearnJCU, under Weekly Labs, Week 2, locate the file "Lab2_if.m". Download this file and save it to your computer.
2. Open Lab2_if.m in MATLAB. Read the code and understand what is happening.
3. To streamline our code, modify it so that only one *if* statement is needed (i.e. merge the two statements into one).

Checkpoint

- Reducing *if* statements, variables and unneccesary code is always a good option when refining code. Remember though, it's best to produce any solution, not a perfect one. Streamlining can always be done later.

Assessable Task 1

Modify Lab2_if.m so that instead of checking if the age is just greater than 18, the age is checked against the following criteria and the corresponding result is displayed. For mastery level students, consider an error check needed to prevent people entering nonsense data such as, -15 or 1530.

- Age < 1 is Baby
- Age 1-13 is Child
- Age 13-18 is Teenager
- Age 18-60 is Adult
- Age >60 is Senior

3 *switch* statement in MATLAB

Terminology

- To instruct a computer to check for equality from a known list of values, you may use a *switch* statement.
- A *switch* statement can only check for equality, not inequality. So relational logic or inequality cannot be used in the cases.
- The syntax of a *switch* statement will vary between programming languages, but the general structure is as follows. Note the use of the *case*, *otherwise* and *end* keywords.

```
switch expression
  case case1
    code to execute
  case case2
    code to execute
  otherwise
    code to execute
end
```

- If none of the cases provided exactly match the expression, the result is otherwise and that portion of code will execute instead.

1. On LearnJCU, under Weekly Labs, Week 2, locate the file "Lab2_switch.m". Download this file and save it to your computer.
2. Open Lab2_switch.m in MATLAB. This code determines, based on a switch variable, a qualitative statement to go with a lab result for EG1002. A simple program like this may be used in something like LearnJCU.
3. Modify this code so that when the switch variable doesn't match one of the switch cases, an error message is issued.

4 *for* loops in MATLAB

Terminology

- To instruct a computer to perform repetitive procedures, a *for* loop can be utilised.
- A *for* loop will cycle over a block of code a specific number of times, each time changing a counter/index variable.
- The syntax of a *for* loop will vary between programming languages, but the general structure is as follows. Note the use of the *end* keyword.

```
for index = values
  code to execute
end
```

- `values` can take three different forms
  - `init:final` - an initial number up to a final number in steps of one
  - `init:step:final` - an initial number up to a final number with the step size step
  - `array` - a prespecified array of values

1. On LearnJCU, under Weekly Labs, Week 2, locate the file "Lab2_for.m". Download this file and save it to your computer.

2. Open Lab2_for.m in MATLAB. Read through the code. Run it and observe the output. Step through the code using the debugging process introduced in Week 1. A better feel for the program flow can be grasped by following step execution in a loop.

Checkpoint

- Write down, in your own words, what this code is doing.
- Recalling from the first lab, why does the sum output to the command line at the end? How would you stop it printing to the command line?

3. What happens if you modify the index (in this case `k`) inside the *for* loop? Will any change to the index be retained in future iterations of the loop?

   (a) After the line, `sum = sum + k;`, insert a line of code to alter the index/counter variable, such as `k=58;`. Run the program. Do we see a disasterous impact on the program?

   (b) Run the program again, but this time enter a breakpoint at the line `sum = sum + k;`. Step through the program using the debug tools to observe the value of `k` in the MATLAB workspace before and after the new line we have introduced.

4. Create an array at the beginning of the program called `values = [1,10,100,1000]` and use this in the *for* loop to find the sum of the first four powers of ten.

Assessable Task 2

(a) Modify Lab2_for.m so that we may compute the sum of even numbers from 1 up to 20. Save this file with a different name. Hint: Consider `index=init:step:final`

(b) Modify your result so that we may compute the sum of even numbers from 1 up to 100 while also computing the sum of odd numbers from 1 to 100.

5 *while* loops in MATLAB

Terminology

- To instruct a computer to repeatedly execute a block of code while a condition remains true, a *while* statement is used.
- *while* loops are commonly used when you do not know how many times you need to loop for, as opposed to a *for* loop where we know exactly how many loops we need to do.
- The syntax of a *while* loop will vary between programming languages, but the general structure is as follows. Note the use of the *end* keyword.

```
while expression
  code to execute
end
```

- Note that the result of a decision is true if the logical operation gives a non-empty array containing only non-zero elements

1. On LearnJCU, under Weekly Labs, Week 2, locate the file "Lab2_while.m". Download this file and save it to your computer.

2. Open Lab2_while.m in MATLAB. **Without running the code**, read it, and determine how many times the program will print to your command line. Once you have an idea of how many times it will print, run the code and see if you were right.

Checkpoint

- What would change if the truth condition was changed to be `n>0` instead of `n>=0`? How many more or less times would the program execute the code within the *while* loop?

3. *while* loops are useful, but they can be troublesome if the condition being tested is ill-planned.

   (a) Change the line `n=n-1;` to `n=n+1;`

   (b) **Before** running the code, determine what will happen as a result of this change.

   (c) Run the code, if you get into trouble press **Ctrl+C**. This keyboard command breaks the MATLAB program execution, which is handy for breaking out of infinite loops (as was the case here).

Assessable Task 3

JCU Engineering Undergraduate Society (EUS) runs a soccer competition among engineering students. The results of the last five years, between 2007-2011, for a postgraduate/staff team playing 10 games a year are `wins=[5,6,7,4,5]` and `goals=[27,34,30,22,29];` as an example, for 2007 they had 5 wins and scored 27 goals.
By using *for* loops, *while* loops and without using built-in MATLAB functions such as *sum, mean*, *mode* and *max* (i.e. by only using your own methods), write a MATLAB script to compute the following:

1. The average number of wins and goals per season?
2. The number of goals per win for each season?
3. Write code to find the largest number in a list of numbers to systematically prove that 2009 had the most wins.

Hints:

- Consider using the *numel* function to determine the number of elements in the `wins` and `goals` vectors.
- Consider using the *zeros* function to preallocate the arrays for your computed results.