Nested if statements

\[
\text{if (Expression1)} \\
\quad \text{Statements1} \\
\text{elseif (Expression2)} \\
\quad \text{Statements2} \\
\text{elseif (ExpressionN)} \\
\quad \text{StatementsN} \\
\text{else} \\
\quad \text{Statements N+1} \\
\text{end}
\]

% EXACTLY 1 of these statements will be executed.

Each Expression is evaluated in sequence, until some Expression is found that is true.
Only the specific Statement following that particular true Expression is executed.
If no Expression is true, the Statement following the final else is executed.
Actually, the final else and final Statement are optional. If omitted, and no Expression is true, then no Statement is executed.

AN EXAMPLE . . .

Example

\[
\text{if (creditsEarned} \geq 90) \\
\quad \text{disp('Senior Status');} \\
\text{elseif (creditsEarned} \geq 60) \\
\quad \text{disp('Junior Status');} \\
\text{elseif (creditsEarned} \geq 30) \\
\quad \text{disp('Sophomore Status');} \\
\text{else} \\
\quad \text{disp('Freshman Status');} \\
\text{end}
\]

Simplify the following code segment, taking out unnecessary comparisons.

\[
\text{if (age} > 64) \\
\quad \text{disp('Senior voter');} \\
\text{end} \\
\text{if (age} < 18) \\
\quad \text{disp('Under age');} \\
\text{end} \\
\text{if (age} \geq 18 \&\& \text{age} < 65) \\
\quad \text{disp('Regular voter');} \\
\text{end}
\]

More examples

average = 100.0;
\[
\text{if (average} \geq 60.0) \\
\quad \text{if (average} < 70.0) \\
\quad \quad \text{disp('Marginal PASS');} \\
\quad \text{else} \\
\quad \quad \text{disp('FAIL');} \\
\quad \text{end} \\
\text{else} \\
\quad \text{disp('FAIL');} \\
\text{end}
\]

% how correct the compile error? Run error?

Do not compare floating-point values for equality. Instead, compare them for near-equality
\[
\text{if (abs(myNumber}-\text{yourNumber}) < 0.00001) \\
\quad \text{disp('Close enough!');}
\]

Conditional Testing and Debugging

- Test data should test all sections of the program
- Use algorithm walk-through to verify that appropriate conditions occur in the right places
- Trace execution of if statements by hand with code walk-through
- Use a debugger to run program in “slow motion” or use debug output statements
Basic Problem Solving with Programming – Iteration (or loops)

What is a loop?
A loop is a repetition control structure

It causes a single statement or block of statements to be executed repeatedly (for a certain number of times or until some event happens)

When use a Loop?
Stated simply, one should use a loop at any point where you wish to repeat a process, idea, or function.

Example: Prompt the user, one at a time, for the high temperature for each day in a week and calculate the average high temperature for the week.

More Loop Examples

Use newton’s method to find the roots (within a certain tolerance) of a real valued function.

Calculate the first 10 terms of the fibonacci series.

Play the game of hangman.

Two types of loops

Count-controlled loops
repeat a specified number of times

Event-controlled loops
something happens inside the loop body that causes the repetition to stop after the current iteration completes

Count-controlled

The problem describes how many times the iteration is to execute
OR
The problem indicates a starting value, ending value and increment (or decrement) for the loop control value (or loop index). The increment does not need to be 1.

Sometime the loop control value (loop index) is needed for some calculation within the block of the loop, sometimes not.
### Event-controlled

- Keep processing until some event happens. This event could be:
  - A special value was entered
  - An end of file was reached when reading a file
  - A certain value was calculated.
- Prompting User Event-Controlled Loop
  - Requires a “priming read”, means you read one data value before entering the event loop.
  - Process data value and then read next value at end of loop, before the next iteration.

### Nested loops

- Some problems require iteration structures to be nested
- Examples:
  - Process data in rows and columns of a table, outer loop is for one row at a time, inner loop is for each column in the current row
  - Outer loop is to allow the user to “play again” or “re-enter data”, inner loop is to do some iterative processing for each outer

### Loop Testing & Debugging

- Test data should test all sections of the program
- Beware of infinite loops - the program doesn’t stop (the event condition to stop the loop is incorrect)
- Trace execution of loop by hand with code walk-through
- Use a debugger to run program in “slow motion”
Write and m-file script in matlab that asks a user what month and day and year they were born and returns their integer age. Hint: c = clock returns a six-element date vector containing the current date and time in decimal form: [year month day hour minute seconds]

Vector problem with conditions on individual elements

Argument Testing EXAMPLES

We have seen how some Matlab functions are allowed to be called with a different number of arguments, and the function executes correctly for that number of arguments. How do we write a function of our own that can accept different number of arguments?

There is a Matlab function you can call within a function, nargin, that returns how many arguments the function was called with.

How can we also tell what data type the arguments passed to are function are? The "isa" function is helpful. http://www.mathworks.com/help/matlab/data-type-identification.html

Add argument checking on a problem from Lab 6 or 7

What is the output of each of these Matlab conditions?

```matlab
x=5; if (x>0) disp('here'); else disp ('there'); end
x=[5 -1]; if (x>0) disp('here'); else disp ('there'); end
x=5; y = -3; if (x>0 && y>0) disp('here'); else disp ('there'); end
x=[5 7]; y=[3 -6]; if (x>0 & y>0) disp('here'); else disp ('there'); end
x=[5 7]; y=[3 -6]; if (any(x>0 & y>0)) disp('here'); else disp ('there'); end
x=[5 7]; y=[3 -6]; if (x>0 | y>0) disp('here'); else disp ('there'); end
```
Two loops in Matlab

while <BooleanExpression>  
  // loop body  
end

for <VariableSpecification>  
  // loop body  
end

% variable specification can be either
  "scalar=vector", iterate through vector
  "scalar=anonymous vector" like 1:10 or 1:2:9

Count-controlled loops contain

• An initialization of the loop control variable
• An expression to test for continuing the loop
• An update of the loop control variable to be executed within each iteration of the body

loopCount = 1; % Initialize loop variable
while loopCount<=10 % Test expression
  % Repeated actions
  loopCount=loopCount+1; % Update loop variable
end

A flag-controlled loop

• Count and sum the first 10 odd numbers in a data file
• Initialize flag notDone to true
• Use while(notDone) for loop test
• Change flag to false when 10 odd numbers have been read or if EOF is reached first

Loops often used to

• Count all data values
• Count special data values
• Sum data values
• Iterate through a vector

Pattern of a nested loop

initialize outer loop
  while ( outer loop condition )
    ...
      initialize inner loop
        while ( inner loop condition )
          inner loop processing and update
        end
    ...
  end

Final Loop Caveat

• You can do many of the same things with vectors and with loops, but in MATLAB, vectors are executed faster
• We will see problems that cannot be solved alone by vectors, and require loops also
Lecture 12 Problems

The following code segment is supposed to write out the even numbers between 1 and 15. It has two flaws in it.

```matlab
n = 2;
while (n ~= 15)
    n = n + 2;
    disp(num2str(n));
end;
```

What is the output of the code as written?
Correct the code so that it works as intended.

Consider the following code segment:

```matlab
sum = 0; i = 1; limit = 8; finished = false;
while (i <= limit & & finished)
    number = input('Enter a value for x: ')
    if (number > 0) sum = sum + number;
    elseif (number == 0) finished = true;
    end;
    i=i+1;
end;
disp(['End of test.' int2str(sum) ' ' int2str(number)]);
```

and these data values are entered one at a time when prompted:

5 6 -3 7 -4 0 5 8 9

What are the contents of sum and number after exit from the loop?

Does the data fully test the code? Explain your answer.
Loop Testing and Debugging

- Test data should test all sections of the program
- Beware of infinite loops -- the program doesn’t stop
- Check loop termination condition, and watch for an OBOB (off-by-1 bug)
- Use algorithm walk-through to verify that appropriate conditions occur in the right places
- Trace execution of loop by hand with code walk-through
- Use a debugger to run program in “slow motion” or use debug output statements

Compare algorithms (vector vs loop)

```matlab
function est=MyPIVector(x,y,n)
hits=sum((x.^2+y.^2)<1);
est=4*hits/n;
end
```

```matlab
function est=MyPILoop(x,y,n)
hits=0;
for i=1:n
    if ((x(i)^2+y(i)^2)<1)
        hits=hits+1;
    end
end
est=4*hits/n;
end
```

Linear Growth – findMax

times = zeros(1,10);
data=randi(1000*(2^10),1,100*(2^10));
for x=1:10 % run ten times for n=200, 400, 800,1600…
n=100*(2^x);
tic
max=data(1);
for i=2:n
    if (data(i)>max) max=data(i);
end
disp(max);
times(x)=toc;
end
x=1:10;
n=100*(2.^x);
plot(n,times,'o');

Quadratic Growth – sort

```matlab
limit=7;
times = zeros(1,limit);
data=randi(1000*(2^limit),1,100*(2^limit));
for x=1:limit % run 7 times for n=200, 400, 800,1600…
    copy=data;
    n=100*(2^x);
tic
    for i=n:-1:2
        max=copy(1);
        for j=1:i
            if (copy(j)>max) max=copy(j);
        end
        copy(i)=max;
    end
    times(x)=toc;
end
x=1:limit;
plot(n,times,'o');
```

Exponential Growth

```matlab
function perm(in, out, used )
    if(length(out) == length(in))
        disp(out);
    else
        for i=1:length(in)
            if (~used(i))
                out(end+1)=in(i);
                used(i) = true;
                perm(in, out, used);
                used(i) = false;
                out=out(1:(end-1));
            end
        end
    end
end
```

Lecture 13 Problems

- Show some loop debugging examples
- Functional Decomposition - break problems into logical pieces that can be solved (programmed) independently
- Solving a continuous problem with discrete approach (approximation approach) – area under a curve
File Reading and Writing

- There are many ways in Matlab to read from and write to files
- The save command is used to save workspace data to a file. See help for more arguments
  ```
  >> save('mySave.mat')
  >> load('mySave.mat')
  ```
- Saving a Session as Text
  ```
  >> diary save.txt
  ```

Reading and Writing from an Excel spreadsheet

```
>>g=xlsread('filename');  % no .xls ending
>>[nums txt raw]=xlsread('filename');  % no .xls ending
```

Reading and Writing from an Excel spreadsheet

- reads `filename.xls` (from the current directory) and places it in an identical array (2-D matrix) inside MATLAB called `g` ASSUMES ALL NUMERIC
- reads `filename.xls` (from the current directory) and places it in an identical array (2-D matrix) inside MATLAB called `g`
- `xlswrite('filename',g);`

Image Processing (matrix approach Data Representation Problem)

A black and white picture can be encoded as a 2D Array

Typical:

- \( 0 \leq A(i,j) \leq 255 \)
- (black) \( \quad \) (white)

Values in between correspond to different levels of grayness.

Just a Bunch of Numbers

```
318-by-250
```

```
48 58 78 67 57 48
62 67 72 72 73 69
102 111 112 112 112 112
157 165 165 165 165 165
196 205 208 207 205 205
199 212 212 212 212 212
190 192 193 195 195 197
174 169 165 163 162 161
```

```
48 55 58 57 57 57
53 67 71 72 72 72
102 111 112 112 112 112
157 165 165 165 165 165
196 205 208 207 205 205
199 212 212 212 212 212
190 192 193 195 195 197
174 169 165 163 162 161
```
A Color Picture is 3 Arrays
Stack them in a 3D array.

Typical:

\[
0 \leq A(i, j, 1) \leq 255 \quad \text{(red)} \\
0 \leq A(i, j, 2) \leq 255 \quad \text{(green)} \\
0 \leq A(i, j, 3) \leq 255 \quad \text{(blue)}
\]

Note 3rd Subscript

Two Problems

We have:

Problem 1
Want:

Problem 2
Want:

Solution Framework

% Read image and convert to a 3D array...
A = imread('IIT_MTCC_Fall.jpg');
% the A matrix contains 8 bit integers

Copy/Manipulate the A array into B array.
(all entries in B must be between 0 and 255)

Convert the B array to a jpg file
imwrite(B,'IIT_MTCC_Fall_NEW.jpg');
The 3D Array

>> [rows, cols, layers] = size(A)
rows = 311
cols = 750
layers = 3
>> % Layers: Red=1 Green=2 Blue=3

For mirror we need to do this for each row for all layers

A  B
1 5
2 4
3 3
2 5
1

New Problem
Color \rightarrow Gray Scale

Take a weighted average of each pixel
0.3*R + 0.59*G + 0.11*B

Produce a Negative from B/W

Lab 10: How would you find the one red tree and change it to black?

See imshow('IIT_MTCC_Fall.jpg') and the data cursor menu option