Math 248: Guidelines for Developing Well-structured Programs

"Structured programming" is the art of writing computer programs that are clear, concise, and accurate. The rules for structured programming are more-or-less universal, the same for every programming language.

Beyond being correct, a well-structured program is characterized by the following attributes:

1. It beings with a PREAMBLE that, at minimum, contains the following on separate COMMENT lines (see example on back):
   (a) Author's name, JMU e-id, and date
   (b) Name of m-file
   (c) Primary purpose of the program
   (d) Major variables and their purposes
   (e) Credits to references or people who have helped the author develop the program
   (f) JMU Honor Pledge (typing "JMU PLEDGE") in the preamble of your program means that you are signing that you have followed the JMU honor code as outlined in the course policies.
2. Descriptive variable names
3. Liberal documentation through the use of COMMENTS
4. Liberal use of "white space" to enhance clarity
5. Indented decision (IF) blocks and loops, with consistent alignment and each level of nesting
6. Careful use of and distinction between REAL and INTEGER variables
7. Legible and aesthetically pleasing output

Programming Tips

1. Start simple and build up. Test as you go.
2. Allow plenty of time - start far ahead of the due date.
3. Write out your program in pseudo-code first.
4. Comment, comment, comment.
5. Allow plenty of time - start far ahead of the due date.
6. Allow plenty of time - start far ahead of the due date.
7. "Simplicity is the greatest sophistication" (Thoreau). Given two equivalent ways to perform a task, always choose the simplest.
8. Allow plenty of time - start far ahead of the due date.

Sample Program [MATLAB]

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NAME: Roger Thelwell
% JMU-EID: thelwerj
% DATE: January 11, 2010
%
% PROGRAM: quadratic_formula.m
%
% PURPOSE:  Compute the roots of a quadratic function
%
% VARIABLES: a,b,c = coefficients of quadratic
%            discriminant = square root of the discriminant, b^2-4*a*c
%            root_1, root_2 = the two roots of the equation
%            real_part1, real_part2 = real parts of each root
%            complex_part1, complex_part2 = complex parts of each root
%
% JMU PLEDGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%\begin{variables}
    a=zeros(1,1); b=zeros(1,1); c=zeros(1,1); two_a=zeros(1,1);
    discriminant=zeros(1,1);
    root_1 = zeros(1,1); root_2=zeros(1,1);
    real_part1 = zeros(1,1); real_part2 = zeros(1,1);
    complex_part1 = zeros(1,1); complex_part2 = zeros(1,1);
%%\end{variables}

%   ask user for input of the coefficients of the quadratic function

    a=input('Please enter the second order coefficient \n');
    b=input('Please enter the first order coefficient \n');
    c=input('Please enter the zero order coefficient \n');

%   compute the roots using the quadratic formula

    discriminant = sqrt(b^2 - 4.0*a*c);
    two_a = 2.0*a;
    root_1 = (-b + discriminant)/two_a;  % compute root 1
    root_2 = (-b - discriminant)/two_a;  % compute root 2

%   separate complex and real parts (only used for the output)

    real_part1 = real(root_1);  % separate real/imag parts for printing (root_1)
    complex_part1 = abs(imag(root_1));
    real_part2 = real(root_2);  % separate real/imag parts for printing (root_2)
    complex_part2 = abs(imag(root_2));

%   outputs roots to the screen

    fprintf(' root 1: %5.2f + %5.2f i \n', real_part1,complex_part1);
    fprintf(' root 2: %5.2f - %5.2f i \n', real_part2,complex_part2);
```