Designing a Program and Subroutines

Summary by: Emmanuel J Rodriguez

Note: Subroutines are commonly called, depending on the programming language, modules, subprograms, methods, and functions.

Top-down design (sometimes called stepwise refinement) is used to break down an algorithm into subroutines.

Top-Down Design Process:

- The overall task of the program is broken down into a series of subtasks.
- Each of the subtasks is examined to determine whether it can be further broken down into more subtasks. This step is repeated until no more subtasks can be identified.
- Once all of the subtasks have been identified, they are written in code.

Three main tools for designing a program and its subroutines:

1. **Hierarchy Chart** – or a structure chart, a top-level visual representation of the main program and the relationships between subroutines.
2. **Flowcharts** – a diagram that graphically depicts the steps that take place in a program.
3. **Pseudocode** – or "fake code" is an informal language that has no syntax rules, it is a "mock-up" program. Each statement in the pseudocode represents an operation that can be performed in any high-level language.

Source: Gaddis, T., *Programming Logic and Design* (New Jersey: Pearson, 2013) pp. 79 – 112.

Top-Down Design
Program: Calculate the average of top eight scores

Overall Task:
Calculate the average of the top eight scores from a list of 20 exam scores.

Scores: 73, 91, 37, 81, 63, 66, 50, 90, 75, 43, 88, 80, 79, 69, 26, 82, 89, 99, 71, 59

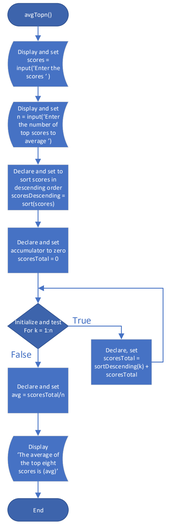
Steps that must be taken to perform the task:
1. Find the top eight scores.
2. Calculate the average of the top eight scores.

1. Hierarchy Chart

Main Program
(Input scores data, Output average of top n scores)

Note: Hierarchy charts does not show the steps that are taken inside a subroutine; they do not reveal any details about how subroutines work.

2. Flowchart



3. Pseudocode

```
% Global variables
scores

%% Main program avgTopn accepts input arguments provided by user, output argument avg
Program [avg] = avgTopn()
    % Display and set scores
    Declare Array scores = input('Enter the scores ')
    % Display and set the top values to average
    Declare Scalar n = input('Enter the number of top scores to average ')

    % Declare and set to sort scores in descending order
    Declare Array scoresDescending = sort(scores, 'descending')

    % Declare and set accumulator to hold a running total. Assigning an accumulator is critical when performing a running total
    calculation, see Gaddis p. 207.
    scoresTotal = 0

        % For loop
        for k = 1:n ; % k is the loop's counter, after every pass through the loop the counter is incremented by 1
            scoresTotal = scoresDescending(k) + scoresTotal ; % scoresTotal on the right hand side of the equation will act as a
            % counter, after each pass the counter variable is assigned to the accumulator variable (they are essentially one in the
            % same), see Gaddis p. 203.
        end for

    % Declare and set average variable
    avg = scoresTotal/n

    % Display the average
    fprintf('The average of the top %f scores is %f, [n, avg])
```