CptS 121 - Program Design and Development

Lab 3: Top-Down Design & Functional Decomposition

Assigned: Week of May 13th, 2019
Due: At the end of the lab session

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Analyze a basic set of requirements for a problem and develop a representative structure chart
- Apply top-down design principles to a problem
- Utilize bottom-up C implementation for a problem
- Identify and implement programmer customized function prototypes, function definitions, and function calls

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

- Analyze a basic set of requirements for a problem
- Declare variables
- Apply C data types and associated mathematical operators
- Comment a program according to class standards
- Logically order sequential C statements to solve small problems
- Compose a small C language program
- Compile a C program using Microsoft Visual Studio 2015
- Execute a program
- Create basic test cases for a program

III. Overview & Requirements:

This lab, along with your TA, will help you navigate through modularizing your C solutions to the provided problems. In this course we will use top-down design and divide-and-conquer strategies to solve problems. Recall that in computer science the concept of top-down design and functional decomposition refers to the process followed to manage the complexity of problems by breaking them down into parts that are easier to understand, implement, and maintain.

Labs are held in a “closed” environment such that you may ask your TA questions. Please use your TAs knowledge to your advantage. You are required to move at the pace set forth by your TA. Please help other students in need when you are finished with a task. You may work in pairs if you wish. However, I encourage you to compose your own solution to each problem. Have a great time! Labs are a vital part to your education in CptS 121 so work diligently.

Tasks: Design, implement, compile, and test C solutions to the following problems. You must use appropriate top-down design and functional decomposition with the problems. Your TA should help you with this. Note: a structure chart is a good way of determining appropriate functions for a problem. You should always have at least the following: one function that gathers input, one function that performs the computation, and one function that displays the result. Once you have completed a problem, demonstrate your solution to your TA. Please collaborate with your teammates throughout solving these problems!

1. For this problem, please create one file (main.c) only. You will apply the 3-file format/organization on the other problems. In lab 2 you should have completed a solution to the following problem (but, this time around rewrite your solution to use functions):

   Write a program to calculate your body mass index (BMI). The BMI is a measurement that uses your height and weight to determine if you are underweight, a healthy weight, or overweight. Your program is required to prompt the user for weight in pounds and height in feet. The height must then be converted to inches (recall: 1 foot = 12 inches). Once the BMI has been calculated display the resultant BMI value. Use the equation below to calculate the BMI.

   BMI = ((weight in pounds) / (height in inches)^2) * 703

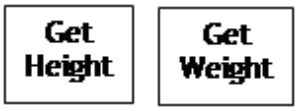
   Note: a BMI of less than 18 indicates you are underweight, >= 18 and < 25 means you are at a healthy weight, >= 25 and < 30 means you are overweight, and > 30 indicates obesity. You do NOT need to classify the BMI value in the program. This would require "if" statements, which you have not learned yet!

   We will practice top-down design in this problem. First, we need to analyze the problem and determine subproblems. Second, we need to draw a *structure chart* to show the relationships between subproblems. The original problem is to compute BMI.

   Original Problem – Level 0

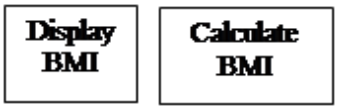
   In order to compute the BMI, we need to get inputs from the user for weight (in pounds) and height (in feet).

   Subproblems – Level 1



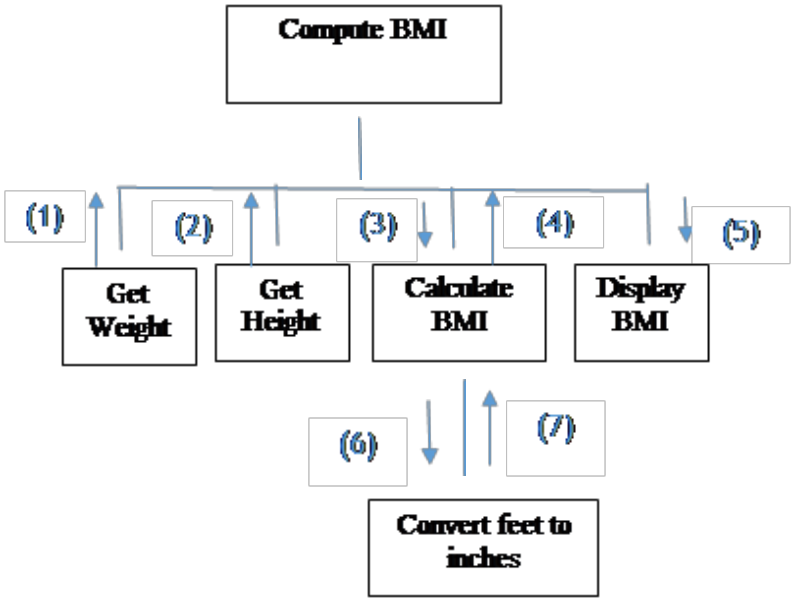
   We also need to convert the height in feet to inches, calculate the BMI based on given user inputs, and display the BMI. Converting height in feet to inches is really a detailed subproblem of calculating the BMI, so it should be placed in level 2 of the chart.

   Subproblems – Level 1 Cont.



   Detailed subproblems – Level 2

The complete structure chart with data flow information is listed below:



Data flow:
(1) Weight in pounds
(2) Height in feet
(3) Weight in pounds, Height in feet
(4) BMI
(5) BMI
(6) Height in feet
(7) Height in inches

You should implement the following functions based on the subproblems:

- double get_weight (void) // prompts the user for weight in pounds, and returns the weight
- double get_height (void) // prompts the user for height in feet, and returns the height
- double convert_feet_to_inches (double height_in_feet) // converts the height of the user from feet to inches, and returns the height in inches
- double calculate_bmi (double weight_in_pounds, double height_in_feet) // evaluates the BMI based on weight in pounds and height in inches, and returns the BMI – call convert_feet_to_inches ()
- void display_bmi (double bmi) // display the resultant BMI value to the tenths place

2. For this problem create three files. Two .c files and one .h file. The files should contain the following:
   - .h file – generally should contain standard library includes, #defined constants, and function prototypes/declarations (we will add more to the .h file in the future!)
   - .c file – contains the definitions for all programmer-defined functions, aside from main ()
   - main.c – contains function main (); recall, we try to design main () to be as concise as possible

   Write a program that computes the duration of a projectile's flight and its height above the ground when it reaches the target. As part of your solution, write and call a function that displays instructions to the program user. You will need to use the following information to solve this problem:

   Problem Constant
   G 32.17 /* gravitational constant */

Problem Inputs
double theta /* angle (radians) of elevation */
double distance /* distance (ft) to target */
double velocity /* projectile velocity (ft/sec) */

Problem Outputs
double time /* time (sec) of flight */
double height /* height of impact */

Relevant Formulas
time = (distance) / (velocity * cos(theta)) /* make sure to include math.h to use cos ( ) and sin ( ) */
height = velocity * sin(theta) * time - ((G * time^2) / 2)

Define functions where appropriate. Recall, building a structure chart is a good way of determining appropriate functions for a problem. Your TA should help guide you with this!

3. Once again, for this problem create three files. Two .c files and one .h file. Write a program that first prompts the user for the scores received on two exams, two labs, and two projects. Note: you should only need to implement one function to get the scores from the user. However, this function will be called six times (once for each score needed). The program must then compute separate averages for the exams, labs, and projects. Note: you should only need to implement one function to compute the average. However, this function will be called three times (once for exams, once for labs, and once for projects). Next, your program must weight the averages according to the following:
   a. Each exam is worth 30%
   b. Each lab is worth 5%
   c. Each project is worth 15%

   Display the weighted average (out of 100%) to the screen.

   Define functions where appropriate!

IV. Submitting Labs:

- You are not required to submit your lab solutions. However, you should keep them in a folder that you may continue to access throughout the semester. You should not store your solutions to the local C: drive on the EME 120/128 machines. These files are erased on a daily basis.

V. Grading Guidelines:

- This lab is worth 10 points. Your lab grade is assigned based on completeness and effort. To receive full credit for the lab you must show up on time and continue to work on the problems until the TA has dismissed you.