CS 135

Design Assignment 3 (DA3-09/27)

Programming Assignment 4 (PA4-10/01)

As specified in your syllabus, you must turn your assignments in by 6:00 pm on the due date specified. If it is turned in late, but prior to 12:00 midnight the day it is due, credit will be reduced by 50% of the earned score. Any laboratories turned in more than 6 hours late will not earn any credit.

Objectives:

1) You will use a set of standardized functions to implement command-line I/O operations in a formatted command line system
2) You will use global constants to assist with program clarity
3) You will use a systematic development process to create a program
4) You will create simple functions as part of applying program modularity, and to allow for reuse of functions when code components are repeated
5) You will implement mathematical operations in order to solve problems

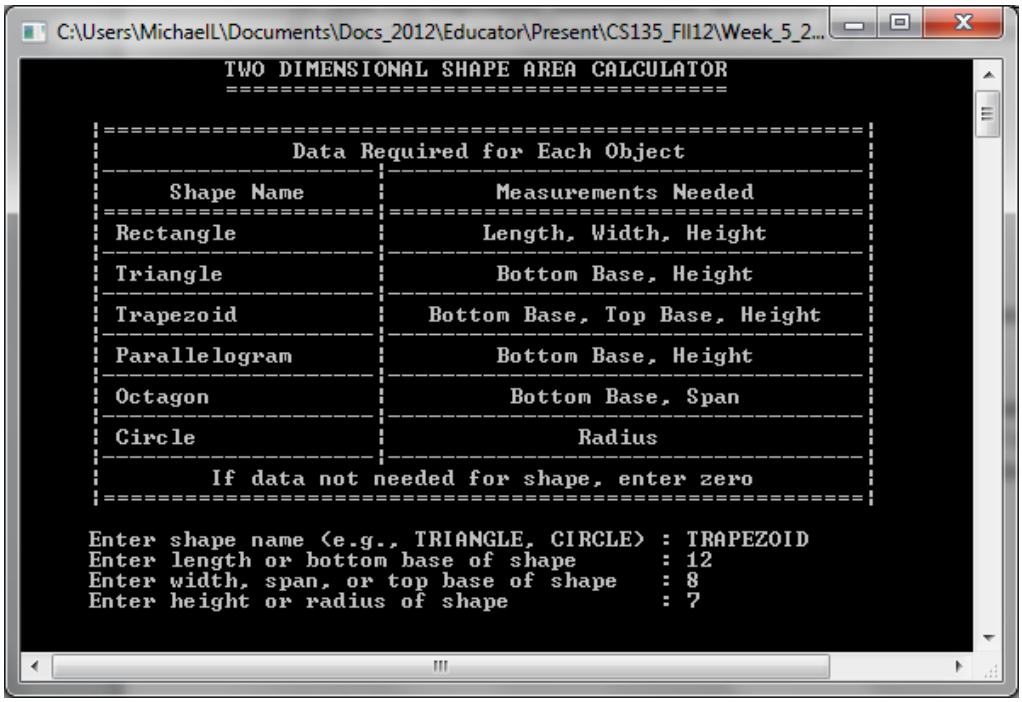
Tasks:

Calculating Two-Dimensional Shape Areas

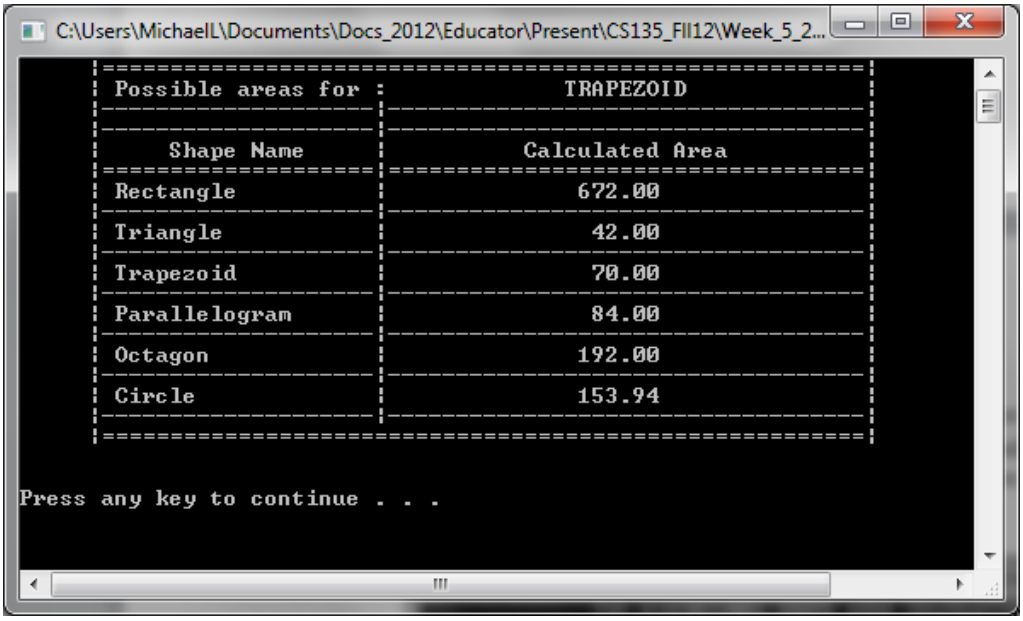
1) The program, called **`shapes.cpp`** will require the use of the **`formatted_cmdline_io_v08.h`** so that nice looking tables can be displayed.

2) The program will accept the kind of two-dimensional shape that is to be calculated and then the data necessary to do the calculations. The output table will show the potential results for all the shapes, but the user will read the line that shows the selected shape.

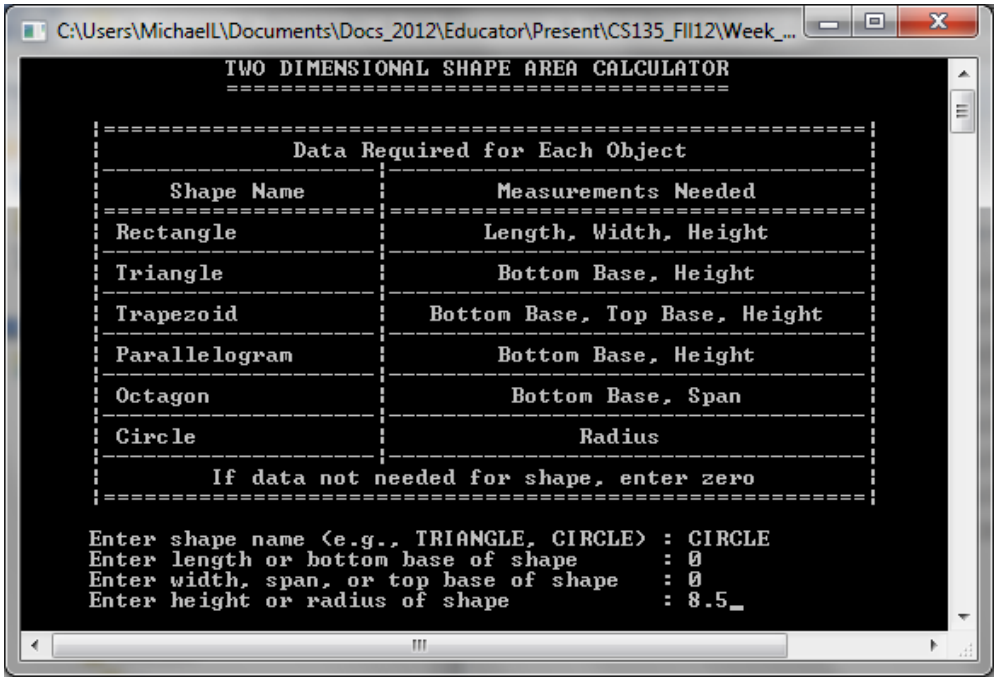
3) The initial part of the program will show a title and it will show the necessary parameters for calculating the areas of one of six shapes. The initial screen and the data entry is shown here. Note that the screen will have to be extended vertically to show all of this part of the program.



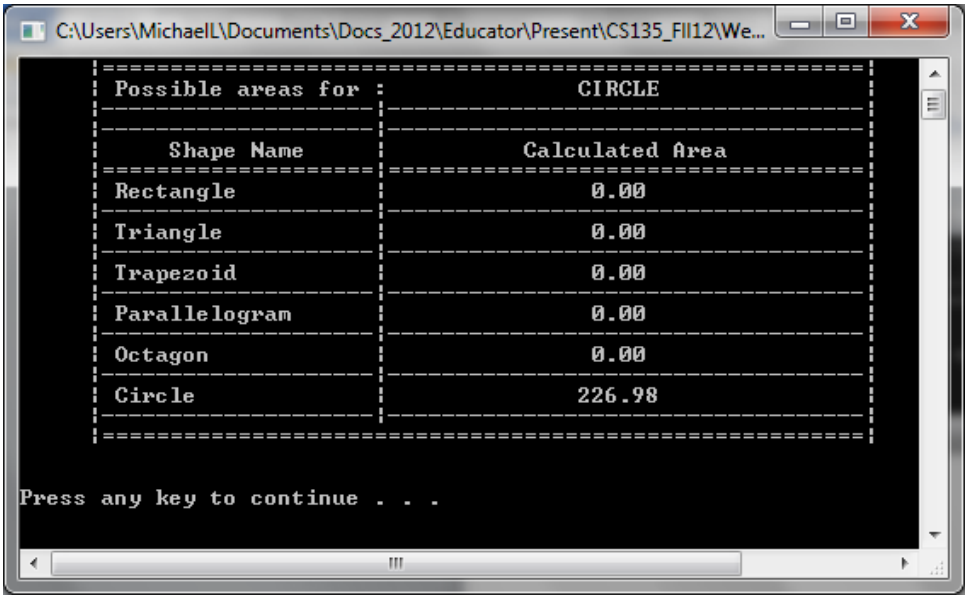
4) Once the third data item is entered, the screen will be cleared and the results will be displayed. Again note that the result table will show more than just the area of the item to be calculated but the user can select the correct area knowing which shape data was entered. The area of this trapezoid is 70.00, as shown here.



5) Some items, such as a circle, need only one length (i.e., the radius). When this is the case, the user enters zero for all the other input items, as shown here.



6) The output for this data set is shown here.



7) You will need to find the equations for each of the area calculations but that should not present any difficulties.

8) This is still only your second program with functions, but again these are all very simple functions. You are ***required*** to create the following functions. You may create others, but be judicious and effective.

   a) **`printTitle`** - prints the initial program title and the underline

   b) **`printRequiredData`** – this is one function that displays the initial table of required input values for each shape. The table, as shown under item 3 previously in this document, has a title, a list of shapes and their required measurements, and instructions at the bottom for entering zero if the measurement item is not appropriate. Use the example program as needed to correctly display this table

   c) **`printOneMeasurementLine`** – this is the function that will display each individual data line within the **`printRequiredData`** function. The **`printRequiredData`** function will first print the main title and subtitles with their boxes, but then 1) one individual shape, 2) its required input values, and 3) the dividing line under it must be printed with the **`printOneMeasurementLine`** function, which will be called six times from the **`printRequiredData`** function. Note that this function will never be called from the **`main`** function, but will called six times from the **`printRequiredData`** function

   d) **`displayResults`** – this function will clear the screen and show the table with the shape name and all the possible areas that the input data would support, obviously including the shape desired by the user. Note: to clear the command-line screen, use the **`system`** function with the command-line function **`cls`** (e.g., **`system( "cls" );`**)

   e) **`printOneResultLine`** – like **`printOneMeasurementLine`** above, this function will be called six times from the **`displayResults`** function to display each shape name, each area calculation result, and then the thin divider line under that displayed result data

   f) **`printThinDividerLine`**, **`printThickDividerLine`**, & **`printThickSolidLine`** – these three functions will be called in several areas in the program to print the solid lines for the tops and bottoms of the tables, and the divided lines as needed within the tables. Since this requires repeated code use, using a function is the best way to respond

   g) Each of the calculations is unique and very simple, so it is not required to create functions for the calculations. However, while it is not required, it would be very good practice for you as you will be developing several processing calculations later on.

9) If you have any questions for following the six step process, review the pertinent sections of the online reference and/or the online code examples and videos, or check with the CS 135 Instruction Team.

10) You will develop a Design Assignment which includes the step 1 through 5 source code files by Thursday at 6:00 pm, and then you must develop the program code for next Monday at 6:00 pm. Further explanation of the required components and uploading process are provided near the end of this document.

11) You must also acquire six screen shots of the input and resulting output of your program operation (two screen shots each for a total of twelve). The first two pairs of screen shots should show the input used and the output displayed in the examples shown previously in this document. The other four should show examples of operations on the remaining shapes (e.g., rectangle, triangle, parallelogram, octagon). Again remember to annotate your screen shot document.

Turning in your Design Assignment:

Information:

Week: 5
Laboratory: 5
Design Assignment: 3
Due Date: 09/27, 6:00 pm

To turn in:

The first five steps of the Six Step Programming Process, including:

1. shape_s1.cpp
2. shape_s2.cpp
3. shape_s3.cpp
4. shape_s4.cpp
5. shape_s5.cpp

Upload these as separate files. Note that following the instructions for uploading your work is critical; you will lose points if you do not follow these instructions. Do not upload any files or in any format other than that specified here.

For information on how to turn in Design Assignments, refer to the "How to Turn in Design Assignments" in the "General Course Information" folder

Important grading note: Due to the large number of students and the small number of people grading assignments, a fraction of the Design Assignments will be graded each week. If your DA is not graded that week, you will receive full credit; if it is graded, you will receive feedback and the appropriate grade. The random process will ensure that all students will be graded an equal number of times across the semester, and it also provides the possibility of getting graded more than one week in a row. You must do your best on each DA and assume you will be graded each week.

Important DA feedback note: If your DA does not get graded in one particular week, you should do the following:

1. At least look at the provided sample DA when it is uncovered on the Tuesday after the PA is due; make sure your structure, organization, and problem-solving strategies are comparable

2. Stop by one of the Instruction Team offices (i.e., Instructors, TAs, Tutors, etc.) and ask them to help you review your DA so you can be sure to do well when you are graded

Turning in your Programming Assignment:

Information:

Week: 5
Laboratory: 5
Programming Assignment: 4
Due Date: 10/01, 6:00 pm

To turn in:

1. The Word file containing the following:

    a. There should be at least six pairs screen shots (i.e., a total of twelve screen shots) for the programs as specified in item #11 previously in this document

    b. Remember to clearly annotate every displayed result

2. The executable file:
    a. shape.exe (shape_s6.exe is acceptable)

3. The source code file:
    a. shape_s6.cpp

These files must be compressed and uploaded as one zip file. To do this, select all of the required files, right click on them, and select “Send To”, then select “Compressed (zipped) Folder”.

Once the folder is created, it will be placed in the same folder in which you are working. Change the name of the zipped folder to “LastnameFirstname_PAX” (where ‘X’ is the number of the Programming Assignment) as shown in the following example: “LeveringtonMichael_PA3” (no quotes). After you have renamed the zipped folder, double click on it to verify that it has all the files it is supposed to have.

Note that following the instructions for uploading your work is critical; you will lose points if you do not follow these instructions. Do not upload any files or in any format other than that specified here.

For information on how to turn in Programming Assignments, refer to the "How to Turn in Programming Assignments" in the "General Course Information" folder