Alleghany International Manufacturing

Cone/Frustum Bulk Data Processing Version

Background

Allegheny International Manufacturing (AIM) senior management is very pleased with the quality of your initial software system. They have requested that the system be modified to also process bulk data. Regional sales representatives save their order information in text files that are consolidated at corporate headquarters. The main task is to modify the previous software application so that it will read sales files, validate entries, perform calculations, and generate summary reports. When the software runs, it shall prompt the user to enter the full path and name of the input data file. Then the software shall open the file, read all of its contents, and perform the required calculations and output.

File Processing

Your software should prompt the user to enter the name and the path to the file containing sales data. Once that information is entered, your software must open the file and process the input data. If the input file fails to open, the software must output an error message indicating a problem occurred opening the file and a notice that processing cannot continue. Abnormal exits are not allowed, use `if/else` structure to ensure that code does not execute if the file fails to open.

If the file opens successfully, the software must read and process each line, or record, in the file. The first line of the file contains column headings. These are for anyone reading the file manually and are not needed for our software. The program should read the entire line of headings to "get it out of the way" and then move on to the second line. The second line of the file is the first record of data that must be processed. We will not know how many total records are in the file. The software must continue reading and processing lines of data until reaching the end of the input file. The file `P2Orders.dat` contains all project data and is available on Blackboard attached to the Project 2 assignment. Each line of the file contains the following data elements:

```
Order Date                  Date of order (string with format yyyy/mm/dd)
Delivery Date               Date of counter due to customer (string with format yyyy/mm/dd)
Order Number                An alphanumeric code (string with no spaces)
State Code                  Stone code (a string containing two digits, no spaces)
Region                      Sales Regions (South, West, North, East, Other) (string, no spaces)
shape Code                  Project 1 Codes (a single character)
color Code                  Project 1 Codes (a single character)
Base Radius (r2)            A floating point number
Top Radius (r1)             A floating point number
Height (h2)                 A floating point number
Base Closed                 Project 1 Codes (a single character)
Top Closed                  Project 1 Codes (a single character)
Customer name & address     Customer's full name and address (string with spaces)
```

Data Validation

The system must test for a variety of possible data errors. You do not need to identify *data type* errors. That means that if a number is expected, then you may assume that the input data file will have a number. If a string is expected, you may assume the input data file will have a string, etc. The possibility of extraneous characters being

present in the data file does not apply. Once you have extracted the value of one data element from the file, you may assume what follows is some amount of blank space followed by the value of the next data element.

As previously mentioned, if the input data file fails to open, then no calculations will be made, and no output (other than an error message) will be sent to the terminal screen. Otherwise, for this version of the project, validation errors will **not** cause the processing to stop. If a data validation check fails, then no calculations will occur for ***that*** record. However, selected values from that row of data will be output along with a list of the errors that were identified (see the sample output presented later). The table below lists the business rules (some new, most from Project #1) that apply to data validation checks. Data items not listed require no validation.

Data Item	Validation Rule(s)
Order Date	Will be in the format yyyy/mm/dd (see notes below)
Delivery Date	Will be in the format yyyy/mm/dd (see notes below)
Shape Code	Must be one of the valid character codes: `C, c, F, f`
Color Code	Must be one of the valid character codes: `R, O, Y, G, B, I, V, r, o, y, g, b, I, v`
Base Radius (r_2)	Minimum 4.0, Maximum 20.0
Top Radius (r_1)	Cone: must be 0.0 Frustum: Minimum $0.5(r_2)$, Maximum $0.75(r_2)$
Height (h_2)	Minimum 5.0, and must be at least equal to r_2; Maximum 25.0
Base Closed	Must be one of the valid character codes: `Y, y, N, n`
Top Closed	Must be one of the valid character codes: `Y, y, N, n`

IMPORTANT NOTES: When reading a date from the file no data validation is required; however, your software should separate the year, month, and day values and store them in integer variables; this applies to the Order Date and the Delivery Date. The year will always have four digits. With respect to the month, the format "mm" means two digits or one digit depending on the month. Similarly, the format "dd" could mean one digit or two digits.

Calculations

Calculations are unchanged from Project 1.

Program Output

When the program runs, it shall output selected values from the input data file. If a record contains errors, also output error message(s). If a record does not have any errors, also output the calculations required for the pattern (line GC, line GF, and angle theta). Keep running totals of the records processed, records with at least one error, cone records without errors, and frustum records without errors. For records without errors, also keep running totals, by shape code, of the total surface area, material cost, and final price. Output these running totals in a summary table after all records have been processed.

Sample output from an execution of the program:

```
------------------------------------------------------------------------------------
                              Shape Orders
------------------------------------------------------------------------------------

Shape  Color                            Base      Top      line      line     angle
Code   Code    r2      r1     h2     Closed   Closed      GC        GF       theta
------------------------------------------------------------------------------------
   C     R   12.48     n/a   12.67      N       Y         n/a     17.78     252.63
   C     I   19.73     n/a   20.70      N       Y         n/a     28.60     248.38
   F     V    5.45    3.40   24.63      N       N       40.99     65.71      29.86
   F     R   17.54    9.95   24.80      Y       N       34.00     59.94     105.35
   C     A    4.43     n/a   24.85      N       Y
           The color code entered is not a valid value.


   F     R   11.82    6.28   24.95      N       Y       28.97     54.53      78.04

           ■■■

   t     Y  113.76   10.87   25.00      N       Y
           The shape code entered is not a valid value.
           The base radius entered is outside of limits.
           The height value entered is not valid.

   C     Y    3.10     n/a   25.00      Y       Y
           The base radius entered is outside of limits.

   F     R    8.19    5.42   25.00      Y       Y       49.22     74.37      39.65
   F     V    5.10    3.41   25.00      N       Y       50.56     75.62      24.28
   C     I   17.50     n/a   25.00      Y       Y         n/a     30.52     206.45

           ■■■

    12971 total records.
      651 records containing at least one invalid value.
     9261 cone records with NO errors.
     3059 frustum records with NO errors.


   Shape         Total            Total            Total
   Code          Area             Cost             Price
-------------------------------------------------------------
     C        16792983.48      80438390.86     101352372.49
     F         6997687.52      33518923.24      42233843.29
-------------------------------------------------------------
 Average          1931.06          9249.78         11654.73
```

How to approach this project

There is no design submission Project #2. You are encouraged to create a pseudocode or flowchart design of the algorithm(s) you plan to implement. The pseudo code terms from Project #1 should be sufficient with the addition of an `ELSE IF` term. For this project several milestones are recommended. You are NOT required to turn anything in or to meet these milestones. However, if you keep to this schedule you should have a good chance of submitting a working program before the deadline. Always make sure that your code compiles and runs before starting the next milestone. Making backups after each milestone is also a good idea.

Milestone 1 – NLT July 19th

- Subscribe to the Project 2 Forum
- Create an empty project: Add skeleton of function `main()` (minimum code should be preprocessor directives, maybe a "bread crumb" or two, and `return 0;`)
- Copy and paste your constants and variables from Project #1

Milestone 2 – NLT July 20th

- Add code to open and read the input data file
- Prompt for the user to enter the full path and name of the input data file
- Read each row into a string variable, output that string back to the terminal window; do not try to store the columns in separate variables yet
- Once you are confident that all rows of the file are being read correctly modify the loop to store the values from each column of the file in a variable of the appropriate type
- Output the selected values (as shown in the sample output of the project description) to the terminal screen, do not worry about exact formatting and alignment yet

Milestone 3 – NLT July 21st

- Add code to output the column headings spaced appropriately
- Add code to format the output data to align neatly under the column headings
- Add code to validate values from the input data file and output error any message(s) below the bad rows
- The validation code should largely be copy and paste from Project #1

Milestone 4 – NLT July 22nd

- Add code to make pattern calculations for records that do not have errors and display those values as shown in the example output in the project description
- Add code make the area, cost, and price calculations and to keep track of all counts and running totals
- Output the counts of records with errors and without errors

Milestone 5 – NLT July 23rd

- Add code to output the summary table of area, cost, and price data
- Verify that your totals and output match that of the example program

Milestone 6 – NLT July 25th

- Make any final enhancements, verify that your program compiles and runs on the server
- Submit your program on time!

Program Source Code

Important: Your output should be very similar to that shown in the sample output, but you do not need to match exactly column-for-column. Do not ask the user for any input other than the data file. This will assist in grading your program.

Include the following comments at the start of your source code file:

```
/*
 * <FileName>.<file extension>
 *
 *  COSC 051 Summer 2015
 *  Project #2
 *
 *  Due on: JUL 25, 2015
 *  Author: <your name>
 *
 *
 *  In accordance with the class policies and Georgetown's
 *  Honor Code, I certify that, with the exception of the
 *  class resources and those items noted below, I have neither
 *  given nor received any assistance on this project.
 *
 *  References not otherwise commented within the program source code.
 *  Note that you should not mention any help from the TAs, the professor,
 *  or any code taken from the class textbooks.
 */
```

These comments must appear **exactly** as shown above. The only difference will be values that you replace where there are "place holders" within angle brackets such as `<netID>` which should be replaced by your own `netID`. For example, I would replace `<netID>P2.cpp` with `waw23P2.cpp`.

Submission Details

Post to Blackboard a `.cpp` file containing your source code. Locate the assignment `Project 2` on Blackboard and attach/upload your file. Do **not** post your executable file. You should ensure that your source file compiles on the server and that the executable file runs and produces the correct output. Use the following file name for your file: `<netID>P2.cpp`

Due date is no later than end-of-day (11:59pm) on July 25th. Late submissions will be penalized 2.5% for each 15 minutes late. If you are over 10 hours late you may turn in the project to receive feedback but the grade will be zero. In general requests for extensions will not be considered.

The value for this project is 100 points.

Grading

	Detailed Rubric (Code)	100.00
1	**Code Quality and Formatting**	***10.00***
	proper indentation	
	good use of constants	
	good use of comments	
	good variable and constant names	
	no abnormal exits	
	proper and consistent indentation	
	good use of horizontal white space to improve readability	
	line length less than 100 characters	
2	**User interface**	***10.00***
	prompts user to enter path and file name for input data file	
	stores user input in variable of the appropriate data type	
	prompt is clear and concise	
3	**File processing**	***20.00***
	file opened correctly	
	test to ensure file opened prior to reading data	
	output the appropriate error message if file does not open, skip all other processing	
	reads all lines of file (no more, no less)	
	close file after all records have been read	
	accommodates upper and lower case character input	
4	**Data validation**	***20.00***
	complete all data validation checks specified in the project description	
	if a record has one or more errors, do not make any calculations, instead output an appropriate error message such as those shown in the project description and example program	
5	**Calculations**	***15.00***
	include calculations for line GC, line GF, and angle theta	
	include calculations for area, cost, and price	
	include calculations for running totals and counts	
6	**Output**	**25.00**
	includes all specified output	
	output is clear and descriptive	
	output is neatly formatted on screen	
	output column headings similar to example program and are neatly aligned	
	output includes appropriate message for invalid data	
	output includes accurate counts and summary statistics as shown in the sample program and screen captures	

	Common Deductions (Code)	
	Program does not compile ON THE CLASS SERVER (deduction varies depending on how bad, value listed is max)	-45.00
	Program compiles but has warnings ON CLASS SERVER (deduction varies depending on how bad, value listed is max	-30.00
	Program crashes during execution ON CLASS SERVER (deduction varies depending on how bad, value listed is max)	-30.00
	Code uses any global variables	-40.00
	Has any abnormal exits (-15 each occurence up to the maximum shown)	-60.00
	Filename does not follow conventions specified	-20.00
	Required comments and honor statement not included at start of file exactly as specified	-30.00
	Late penalty for each 15 minutes late	-2.50

Academic Integrity

This is an individual project and all work must be your own. Refer to the guidelines specified in the *Academic Honesty* section of this course syllabus or contact me if you have any questions.