

Project #2 – Lies, Damn Lies, and Statistics

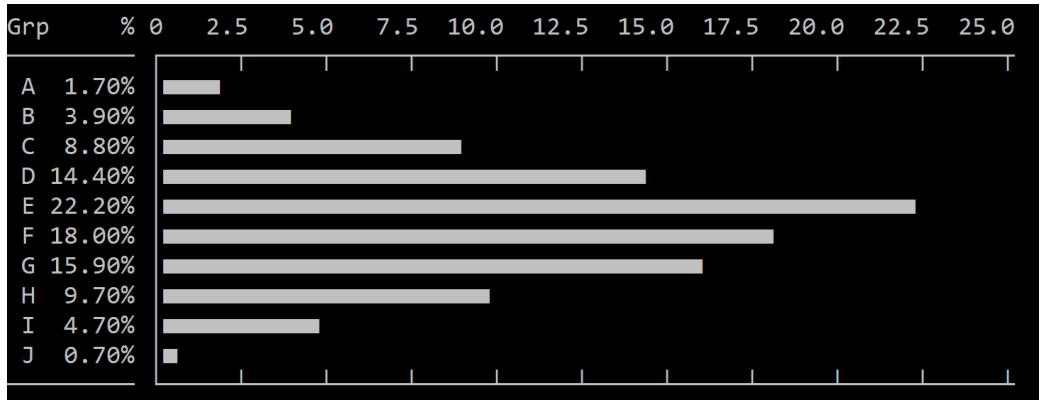
Course	INFO-1156 Object-Oriented Programming in C++
Professor	Garth Santor, and Lynn Koudsi
Assigned	Tuesday, June 1 st , 2021
Due	Monday, Jun 28 th , 2021 by 11:59 pm
Weight	9%

Project Description

Create a C17 console application to compile the following statistics on a list of real numbers:

For the following formulae x is the datum, X is the dataset, where $x \in X$ (x is an element of X).

Metric	Formula
Number of values	<i>The count of the elements in the dataset.</i> $ X $
Minimum value	<i>The value that is less than or equal to all values in the set.</i> $\min \leq x, \forall x \in X$
Maximum value	<i>The value that is greater than or equal to all values in the set.</i> $\max \geq x, \forall x \in X$
Range	<i>The range of values of the dataset.</i> $\max - \min$
Statistical median	<i>The middle value of a sorted data set of odd length, or the arithmetic mean of the two closest values to the middle of a sorted data set of even length.</i> $Y = \text{sorted}\{x_1 \cdots x_n\}$ $\tilde{x} = \begin{cases} Y_{(N+1)/2} & \text{if } N \text{ is odd} \\ \frac{1}{2}(Y_{N/2} + Y_{1+N/2}) & \text{if } N \text{ is even} \end{cases}$
Arithmetic mean	<i>The sum of all the values divided by the number of values.</i> $\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$
Variance	<i>The mean of the squared differences of each sample from the arithmetic mean.</i> $\text{variance} = s_N^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$
Standard deviation	<i>The square root of the variance.</i>

	$\text{standard deviation} = s_N = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2}$																						
Frequency table	<p><i>A table listing the interval, and the count of data set values in that interval, and the percentage of the dataset represented by count of that interval.</i></p> <p><i>Our frequency table will use ten buckets.</i></p> <pre> A [14.0 .. 40.9) = 17 : 1.70% B [40.9 .. 67.8) = 39 : 3.90% C [67.8 .. 94.7) = 88 : 8.80% D [94.7 .. 121.6) = 144 : 14.40% E [121.6 .. 148.5) = 222 : 22.20% F [148.5 .. 175.4) = 180 : 18.00% G [175.4 .. 202.3) = 159 : 15.90% H [202.3 .. 229.2) = 97 : 9.70% I [229.2 .. 256.1) = 47 : 4.70% J [256.1 .. 283.0] = 7 : 0.70% </pre>																						
Histogram	<p><i>A graphical depiction of a frequency table – typically a horizontal bar chart.</i></p> <p><i>Labels indicate percentages.</i></p>  <table border="1"> <thead> <tr> <th>Grp</th> <th>%</th> </tr> </thead> <tbody> <tr><td>A</td><td>1.70%</td></tr> <tr><td>B</td><td>3.90%</td></tr> <tr><td>C</td><td>8.80%</td></tr> <tr><td>D</td><td>14.40%</td></tr> <tr><td>E</td><td>22.20%</td></tr> <tr><td>F</td><td>18.00%</td></tr> <tr><td>G</td><td>15.90%</td></tr> <tr><td>H</td><td>9.70%</td></tr> <tr><td>I</td><td>4.70%</td></tr> <tr><td>J</td><td>0.70%</td></tr> </tbody> </table>	Grp	%	A	1.70%	B	3.90%	C	8.80%	D	14.40%	E	22.20%	F	18.00%	G	15.90%	H	9.70%	I	4.70%	J	0.70%
Grp	%																						
A	1.70%																						
B	3.90%																						
C	8.80%																						
D	14.40%																						
E	22.20%																						
F	18.00%																						
G	15.90%																						
H	9.70%																						
I	4.70%																						
J	0.70%																						
Logarithmic transform	<p><i>A logarithmically transformed dataset is defined as: $x' = \ln x \forall x \in X$</i></p> <p><i>This means that the transformed dataset has the natural logarithm of each datum in the original dataset.</i></p> <p><i>Provide both the frequency table and histogram of the log-transformed data.</i></p>																						

Your program must handle any length of list (**potentially billions!**) The list will be input (or piped) from the console, or read from a file. The list is terminated with *end-of-stream* (^Z, or ^D on Linux) or the word *end*. Bad input is to be skipped. Sample input lists are posted on FOL.

Keep the output clean and minimal. A sample output file is posted on FOL.

You will have to sort the data to compute the median and are to implement your own sort. Your grade is dependent upon the sorting algorithm that you choose (to get full marks, implement the quick, merge sort or heap sort). Sorting algorithms can be found at [GATS Companion to Searching and Sorting](#).

A sample of the executable, input file format and output file format are posted on FOL. Your output should be formatted as the output file. All statistics numbers calculated should be displayed to 3 decimal places, where applicable. You should create more input files to verify that all your statistics are done properly.

Grading Criteria

Input Requirements	Weight	Points	Awarded	Grade
Reads double precision real numbers	1%	1	1	1%
Terminates input on EOF	4%	1	1	4%
Doesn't handle any size data set (array is NOT heap allocated)	-25%	1		0%
Skips bad inputs (only terminates on 'end' or EOF)	5%	1	1	5%
Input from keyboard	1%	1	1	1%
Input from file named on command line	5%	1	1	5%
Reports bad filename	4%	1	1	4%
Statistics Requirements				
Number of values reported	2%	1	1	2%
Minimum value correctly reported in all cases	2%	1	1	2%
Maximum value correctly reported in all cases	3%	1	1	3%
Range value correctly reported in all cases	3%	1	1	3%
Mean correctly reported in all cases	5%	1	1	5%
Median correctly reported in all cases	5%	1	1	5%
Population variance correctly reported in all cases	5%	1	1	5%
Population standard deviation correctly reported in all cases	5%	1	1	5%
Logarithmic transform	5%	1	1	5%
Frequency Table Requirements (must complete statistics first)				
Group labels	2%	1	1	2%
Minimum and Maximum ranges	5%	1	1	5%
Count is displayed and correct	5%	1	1	5%
Percentage is displayed and correct	5%	1	1	5%
Histogram Requirements (must complete Frequency table first)				
Group labels	1%	1	1	1%
Percentages rounded to 2 decimal places	1%	1	1	1%
Bars are correct length	2%	1	1	2%
Bars scales are correct	3%	1	1	3%
Bar scales are adjusted to next highest multiple of 5% greater than maximum percentage	3%	1	1	3%
Non-functional requirements				
Sort is: bubble	0%	1		0%
Sort is: insertion sort	5%	1		0%
Sort is: <stdlib.h> qsort	2%	1		0%
Sort is: quick/merge/heap	10%	1	1	10%
Sort is: Bogo/Bozo	-5000%	1		0%
Labels are not left justified	-5%	1		0%
Values are not right justified	-5%	1		0%
Values are not rounded to 3 decimal places	-5%	1		0%
Frequency table code is contained in a function	3%	1	1	3%

Histogram code is contained in a function	2%	1	1	2%
Multi-file solution	3%	1	1	3%
Penalties				
Penalties from <i>C & C++ Grading Guide v2.3.0</i>	-5%	1	0	0%
Late submission:	-10%	1	0	0%
Total				100%

Submission Requirements

1. Submit **entire Visual Studio project directory** to Fanshawe Online
 - a. Delete ***all*** debug and release directories.ⁱ
 - b. Submit in a .ZIP, .7z archive file.

ⁱ Alternatively, you can ‘clean’ your project for submission by downloading ‘vsclean’ a Visual Studio Solution Cleaner from www.gats.ca.