SEP105: Introduction to Programming for Engineers

Project 3

T2 2025

Due Date: See the unit site

Written by: Dr. Benjamin Champion

Project 1 – Description

The Turing Moore Engineering company has tasked you writing a program to do an analysis of the local water sources.

As a minimum, the program must display a Welcome message as the first thing displayed in the program with the following information, each on a separate line:

- The message “Turing Moore Water Safety Calculator 1.0”
- Your name (as shown on the unit site)
- Your Student ID
- Date the assignment is due in the format: dd/mm/yyyy
- The highest part your program has attempted (part 1, 2, 3 or 4)

An example of what this **may** look like is shown below:

Turing Moore Water Safety Calculator 1.0
Name: Benjamin Champion
Student ID: 123456789
Completed Up To: Part 2

The following points apply to this document:

- The entire assessment is worth 40% of the total grade.
- The program must be written entirely by you!
- The flowchart must be submitted as a .svg file
- The program must be written in C++ targeted at the Windows Operating System. Please use Visual Studio Code to write and compile your code. If you are using a non-windows machine (eg MAC) you will need to compile and test your code on a Windows machine (or virtual machine) before submitting
- Only functions, types, etc that have been covered in class between **weeks 1-11** can be used in this assessment. Anything else being used may be considered for academic misconduct. See the Project 3 – Allowed Variable Types and Functions document on the unit site for a comprehensive list of what is allowed.
- All files used in this project must be submitted to the dropbox as a single .zip file, with the file name having the following format:

  studentID_FirstName_LastName_AssignmentName

  For example, an assignment file name might look like:

  123456789_Benjamin_Champion_Project3.zip

The following information describes the requirement for the different levels in this assignment. Before completing tasks from a higher level, all tasks from a lower level must first be completed.

Presentation (Code Defence)

Hurdle – no marks, must pass to pass the unit

You must pass a defence of your code to pass the unit. The code defence will be around 5 minutes, during weeks 11 or 12 (as time permits). All code defence will be online via zoom and will be recorded. Check the unit site for your allocated time close to week 11. Make sure you have some photo ID (Deakin University student ID is preferred, we will also accept a passport, Australian drivers' licence or Australian Proof of Age card) to prove your identity during the code defence.

For the code defence, you should not need to do any preparation! You do **not** need slides, a speech, etc. The assessors will bring up the code you submitted to the drop box and ask you a couple of questions about your code which you must answer. It is highly recommended you also have your code open in front of you, (please use the **EXACT** copy you submitted to the drop box) as we may reference specific line numbers. This is your chance to convince the assessors that you wrote and understand your code! If you get a question wrong, stumble around an answer, have a mental blank, etc do not stress. We understand this is the first experience of this kind for many students.

Hint: If you have been using AI to write most of your code, this will be very difficult! If you have written the code yourself, the defence should be much easier.

Flow Chart

50% of the mark

Create a flow chart to plan and represent your program. It is highly recommended that you make the flow chart **BEFORE** writing any code as it will help you plan out your code. Make sure you follow the structure, symbols, etc set out in Week 1.

Programming problem

50% of the mark

Download the data sets from the unit site, extract the .zip. Place the Water_Temperature.csv, Water_Course_Level.csv and Water_pH.csv from the clean folder in the same folder as your main.cpp file (unless directed otherwise). All questions asked in all parts must be solved by your program with the data from these files. Files with the same format, but different data, may be used when marking the assessments.

Any reference to files in your program must be completed using relative pathing. Absolute pathing will attract a significant reduction in marks as the program will not work on the assessors PC!

Part 1:

- Prompt the user with a formatted message with instructions on how to use the software
- Open Water_Temperature.csv and Water_Course_Level.csv
- Find the maximum value in each data set

- Find the minimum value in each data set
- Find the average value for each data set
- When processing the temperature data, ask the user if they would like the value in degrees Celsius or degrees Fahrenheit. All values should be converted into the desired form before any operations are performed.
- Display the parameter type, station long name and station number to the terminal immediately before displaying any calculated data for that data set.
- Display values found for the entire data set with the date (yyyy-mm-dd), the value that was recorded and an appropriate label. For example, if the maximum value was found to be 3.2 and recorded on the 21/2/2024 it should be displayed to the terminal like:

  Maximum Value: 3.2, Date: 2024-02-21

  If a value does not have a date, for example the average, the date can be omitted.
- All information written to and received from the terminal should be written to a file called log.txt. This file should contain all data that was written to the terminal during the programs run, and all data the user entered into the program (even if it was an error) during the programs run.
- When the user enters anything into the program, you must make sure their input is valid, and if not display an appropriate error message and ask for the value again.
- At the end of the program, prompt the user if they wish to exit. If they do, exit the program. If they don't, start the program again.
- All values displayed to the terminal should be to three decimal places. Values do not have to be rounded.

Part 2:

In addition to all of part 1, the program must:

- Find the month with the highest average value in each data set
- Find the month with the lowest average value in each data set
- Find the median value for each data set
- Find the mode of each data set
- Find the month with the most consistent values for each data set. The month with the most consistent values will be the one with the lowest variance.
- Add the appropriate unit to the data being printed out. If the unit is not a standard ASCII character, for example there is no degree symbol (°), use text instead (for example deg).
- Display values found for a specific month including the name of the month, the year the month occurred, the specific value and an appropriate label. For example, if the month with the highest value was January, its value was 20.3 and the year it occurred in was 2024 it should be displayed to the terminal like:

  Highest Average Value for a Month: 20.3, Month: 01, Year: 2024

  Do not hardcode the year! The test file may have a different year as from the provided data files!
- Create a new file called Monthly_Values.csv. Format this file so it clearly displays:
  - Average value for each month
  - Lowest value for each month
  - Variance calculated for each month
  - Any other data that may be asked to calculate or display for a month

Part 3:

In addition to all of part 1 and part 2:

- Sometimes, data files are missing data. Your program should now work with the files in the corrupted folder as well as files in the clean folder. If a file is missing some information on a particular row, that entire row should be disregarded from all forms of data processing.
- Find the standard deviation for each data set.
- Display how many values are within 1 standard deviation from the mean for each data set.
- Display how many days are within 1 standard deviation of the mean at the same time in all three data sets.
- Ask the user to enter in a minimum and maximum value threshold value for each data set. Display the number of times each data set crossed each threshold value.
- Create a new .csv file called Cleaned_Data. If the files in the corrupted folder are used, the missing entries for a particular data set should be blank for that day. This file should contain information from all three data sets with the following headings:
    - Date
    - Water Course Level
    - Water Temperature
    - Water pH
    - Min Threshold Reached – Water Course Level
    - Min Threshold Reached – Water Temperature
    - Min Threshold Reached – Water pH
    - Max Threshold Reached – Water Course Level
    - Max Threshold Reached – Water Temperature
    - Max Threshold Reached – Water pH

Part 4:

In addition to all of part 1, part 2 and part 3:

- Calculate and display the covariance between the water temperature and the water course level.
- Calculate and display the covariance between the water temperature and the water course pH.
- Display both of the covariance levels to the terminal
- Depending on the covariance levels display the two most appropriate messages:
    - In general, the water course level increases as the temperature increases
    - In general, the water course level decreases as the temperature increases
    - In general, the water pH increases as the temperature increases
    - In general, the water pH decreases as the temperature increases
- Move the data sets into a folder called data, and download data from 5 different sites. The website, and parameters used to download the data, can be found in the Hints section. At the start of the program, prompt the user to select which site they wish to analysis. Include the site names and numbers, read from the files (not hard coded) when prompting the user. All analysis should be conducted on the data set selected by the user
- Make a new folder in the data folder containing the same site files, but with data from the last 10 years. Before prompting the user to select a site, prompt the user if they wish to analysis the data from 2024 or the last 10 years. Depending on there answer, will depend on which files are used for analysis. If the last 10 years is selected, any calculation or display to do with a month in the data set (eg the month with the highest average value) should also include the

year. In this way, the data from a month in 1 year should not effect the data in the month of a different year.

Appendix

Prior knowledge

At a **minimum** to pass this assessment you will need to have skills in the following areas. If you feel like you are not as strong in these areas as you would like to be, I recommend you go and brush up on the class content. Please note, depending on how you write your program, and the parts you complete, you may not use everything listed. That is OK as there are always multiple ways to write a program with the same functionality.

- Flow Diagrams
- Initialising and manipulating variables (int, float, double, etc)
- Reading and Writing from the console
- Mathematical Operations
- Conditionals (&&, ||, etc)
- Decisions (if, switch)
- Debugging
- Looping (for, while, do while)
- Arrays
- Structures
- Vectors
- Pointers
- Functions
- Classes
- File IO
- Custom Libraries
- Lists

What to submit

A .zip folder containing the following items:

- The flow chart as a .svg file
- All fully commented code files that pertain to the project (eg the main.cpp file). Your submitted file(s) must be the files used to compile the executable.
- An .exe of the final code. This .exe must be compiled to run on a Windows PC. It is up to you to ensure (and test) that the software that you write will execute correctly on a Windows PC. If you use a Windows PC, and the methods that are described in class, the code will run and execute on another Windows PC. If you use some other operating system (eg MAC, Linux, ChromeOS) the executable file will not run on a Windows PC. If this is you, borrow or emulate a computer running the Windows operating system to compile your code before submitting it. Consider this your only warning, any software that does not run for this reason will generate 0 marks for the appropriate sections of the rubric.
- The entire project folder containing the software. This is independent to the other documents you are required to submit but should be in the same .zip folder

Hints

- It is recommended that you develop the program in stages, do not shoot for a High Distinction straight away. First make a flowchart and program that will complete part 1. Save this, make a new program, copy in the previous solution and modify the flow chart and program so it can complete parts 2, etc. Make sure you have all of the functionality working of the previous part **BEFORE** moving onto the next part. No marks will be awarded for functionality for higher parts until all previous parts have been fully completed. You should only submit the most complete working version of your program.
- Make your flowchart(s) **BEFORE** you write any software. This will help you reduce the amount of code you need to write and help you significantly in debugging your code!
- While it is tempting to use AI such as Chat GPT to help write your code, in this unit we recommend against using it for your assignments. We are fully aware that we cannot stop you! While AI is very good at writing code, you are not going to practice, and learn, the fundamentals of code. If you rely on AI to write your code, you will also find the defence in the final assessment much, much harder as you will not be able to answer our questions, possibly causing you to fail the hurdle, and therefore the unit!
- The data files for this assessment are taken from: http://www.bom.gov.au/waterdata/, selecting station number 233217 for the year of 2024 with a Daily mean time series. The Water Course Level, Water pH and Water Temperature Data Sets were used. It is highly recommended that you try different site numbers to validate your code on different data sets. You may need to rename the files once downloaded, and place them in the same location, to make them work with your code.

Engineering Notation

In Engineering it is common to use engineering notation to display units. Some commonly used engineering notation is:

Notation	Symbol	SI equivalent
pico	p	10^{-12}
nano	n	10^{-9}
micro	µ	10^{-6}
milli	m	10^{-3}
kilo	k	10^{3}
mega	M	10^{6}
giga	G	10^{9}
terra	T	10^{12}

Required Theory

Degrees Fahrenheit to Deg Celsius:

$$°F = 32 + \frac{9}{5}°C$$

For example, if it is 25 degrees Celsius:

$°F = 32 + \frac{9}{5}°C = 32 + \frac{9}{5}25 = 77$

Mean

The mean of a data set can also be called the average of a data set and is represented by:

$$mean = \mu = \frac{1}{N}\sum_{n=0}^{N-1} x_n$$

Where x_n is the n[th] value in the set, and N is the number of values in the set. For example, if we had the numbers:

1, 8, 5, 3, 26, 4, 7

The mean would be:

$\mu = \frac{1}{N}\sum_{n=0}^{N} x_n = \frac{1}{7}\sum_{0}^{6} x_n$
$\mu = \frac{(1+8+5+3+26+4+7)}{7}$
$\mu = 7.71$

Median

The median is the middle number of a set of data. The median value is sometimes used over the mean as it is not affected by outliers in the data set. To find the median value, simply order the data set from largest to smallest, and pick the middle number (the value where there is an even group of numbers to the left and right of the value). For example, if we use the same numbers:

1, 8, 5, 3, 26, 4, 7

First, you order the numbers:

1, 3, 4, 5, 7, 8, 26

And then pick the middle number (so there is the same number of numbers on both the left and the right):

$median = 5$

If the total number of elements is even, then there is no middle number as the set can be broken into two even groups. In this case, the two middle numbers are chosen and averaged to find the median. For example, if we had this set of numbers:

5, 4, 6, 3, 8, 7

First, order the numbers:

3, 4, 5, 6, 7, 8

Find the middle numbers. In this case, we have two, so they are:

5 and 6

Now, find the average between the numbers:

$median = \frac{5+6}{2} = 5.5$

Mode

The mode of a data set is simply the number(s) that appear the most in the data set. It is possible to have multiple modes in the data set, or to have none at all (assuming all numbers only appear once or an equal number of times).

For example, given the following data set, find the mode(s):

1, 8, 1, 7, 3, 6, 2, 4, 1, 9

In this case, 1 appears 3 times. This is the most times, so it is the mode of the data set.

Lets look at another example. Consider this data set:

2, 8, 6, 4, 2, 5, 9, 7, 2, 4, 6, 4

In this case, 2 appears three times and so does 4. Therefore, they are both the mode of the data set.

Variance

The variance can be used to compare different data sets to each other to determine how close on average the values in the data set are to each other. This should mean that a data set with values that are generally closer to each other should have a lower variance than a data set with values that are on average a longer way apart from each other.

Calculating the variance is relatively straightforward. We can use the following equation:

$$\sigma^2 = \frac{1}{N}\sum_{n=0}^{N-1}|x_n - \mu|^2$$

Where σ^2 is the variance, x_n is the nth point in the set, μ is the mean of the set and N is the total number of elements in the dataset.

For example, let's consider the following data set:

1, 3, 5, 7, 14

First, calculate the mean of the data set:

$\mu = \frac{1}{N}\sum_{n=0}^{N-1} x_n = \frac{1}{5}\sum_{0}^{4} x_n$
$\mu = \frac{(1+3+5+7+14)}{5}$
$\mu = 6$

Next, can take each value in the set away from the mean and square it:

Value (x)	$\|x - \mu\|^2$
1	25
3	9
5	1
7	1

14	64

Finally, sum them all together to get the variance:

$\sigma^2 = \frac{1}{N}\sum_{n=0}^{N-1}|x_n - \mu|^2$

$\sigma^2 = \frac{25+9+1+1+64}{5}$

$\sigma^2 = 20$

Standard Deviation

The standard deviation can be used to measure how clustered data is around the mean. The larger the data is spread out, the larger the standard deviation will be! To calculate the standard deviation, the following equation is used:

$$SD = \sqrt{\sigma^2}$$

Where x is the current value, σ^2 is the variance of the data set.

For example, let's consider the following data set:

8, 5, 8, 4, 3, 2, 6, 8, 9

First, we need to find the mean of the data set:

$\mu = \frac{1}{N}\sum_{n=0}^{N-1} x_n = \frac{1}{9}\sum_0^8 x_n$

$\mu = \frac{(8+5+8+4+3+2+6+8+9)}{9}$

$\mu = 5.89$

Next, we can find the variance of the data set:

Value (x)	$\|x - \mu\|^2$
8	4.45
5	0.79
8	4.46
4	3.57
3	8.35
2	15.12
6	0.01
8	4.46
9	9.68

$\sigma^2 = \frac{1}{N}\sum_{n=0}^{N-1}|x_n - \mu|^2 = 5.65$

And finally, we can calculate the standard deviation:

$SD = \sqrt{\sigma^2}$

$SD = \sqrt{5.65}$

$SD = 2.38$

To find a percentage of how many values fall within one standard deviation of the mean simply order the data set:

2, 3, 4, 5, 6, 8, 8, 8, 9

Calculate a value either side of the mean:

Lower limit: $\mu - SD = 5.89 - 2.38 = 3.51$

Upper limit: $\mu + SD = 5.89 + 2.38 = 8.27$

Then count the values that fall between the Upper and Lower limit. In this case, we have 6 values (4, 5, 6, 8, 8 and 8) that fall within one standard deviation of the mean, or 66.7% of values. Generally, this analysis is performed on data sets much larger than this to show any real benefit.

Covariance

The covariance between a data set is a measure the describes the direction of a linear relationship between two variables, (eg data sets). It indicates if two things tend to go up (positive covariance) together or tend to go down (negative covariance) together.

The covariance can be described by:

$$Cov(X,Y) = \frac{1}{N}\sum_{n=0}^{N-1}(x_n - \mu_x)(y_n - \mu_y)$$

For example, lets consider the two data sets:

X	Y
20	3.2
22	3.4
21	3.5
23	3.9
24	4.0

First, calculate the mean for each set:

$\mu_x = \frac{1}{N}\sum_{n=0}^{N-1} x_n = \frac{1}{5}\sum_{0}^{4} x_n = 22$

$\mu_y = \frac{1}{N}\sum_{n=0}^{N-1} y_n = \frac{1}{5}\sum_{0}^{4} y_n = 3.6$

Next, compute each term:

$(x_n - \mu_x)$	$(y_n - \mu_y)$	$(x_n - \mu_x)(y_n - \mu_y)$
-2	-0.4	0.8
0	-0.2	0
-1	-0.1	0.1
1	0.3	0.3
2	0.4	0.8

Sum them all together:

$\sum_{n=0}^{N}(x_n - \mu_x)(y_n - \mu_y) = 0.8 + 0 + 0.1 + 0.3 + 0.8 = 2.0$

Finally, divide by the total number of samples:

$Cov(X,Y) = \frac{1}{N}2.0 = \frac{2.0}{5} = 0.4$

Rubric

All Assignments will be marked off the following Rubric. Please note:

- No marks will be awarded for a higher part, if the lower part is not completed. For example, if you get half of part 2 working, and some of part 3, the part 3 components will not be assessed as part 2 was not completed.
- The flow chart must represent the code that was submitted. Do the flowchart before writing any software! It will help you iron out many bugs early, before you start writing any of the code. If the flow chart includes some functionality that was not included in the final solution, this is OK. Code that is not represented in the flowchart should be avoided and may be penalised.
- The submitted code must be the same as the executable file that was submitted. If this is not the case, ie the functionality of the code does not match what is observed in the executable, you may be subject to academic misconduct.

See next page for the rubric

	HD	D	C	P	F
Flow Chart Presentation	The flow chart is laid out exceptionally well. Is very easy to read, understand and follow all aspects of the flow chart. The correct symbols are used for all or most sections of the flow chart. All arrows are labelled where appropriate.	The flow chart is laid out exceptionally well. Is very easy to read understand and follow most components of the flow chart. The correct symbols are used for all or most sections of the flow chart. All arrows are labelled where appropriate.	The flow chart is laid out in a way that makes sense. Is easy to read, understand and follow most aspects of the flow chart. The correct symbols are used for all or most sections of the flow chart. All arrows are labelled where appropriate.	The flow chart is laid out in a way that makes sense. The information in the flow chart can be read, makes sense and the flow chart can be followed. The correct symbols are used for all or most sections of the flow chart. All arrows are labelled where appropriate.	Either the flow chart is missing some information, is laid out in a way that is difficult to follow, cannot be read, consistently uses the wrong symbols or does not label all arrows when appropriate.
Flow Chart Function	The program demonstrates some or all functionality for all parts. The function of all components make sense.	The flow chart demonstrates some or all functionality for parts 1, 2 and 3. The function of all components make sense.	The flow chart demonstrates some or all functionality for part 1 and 2. The function of all components make sense.	The flow chart demonstrates all of the functionality of part 1. The function of all components make sense.	The flow chart either is not present, does not show enough information or functionality for part 1, or does not resemble the functionality of the submitted code.
Comments	All aspects of the code are fully commented. This includes all header blocks, etc where appropriate. All lines of the code have a comment describing their purpose, where appropriate, and is not simply a redefinition of the code.	All aspects of the code are fully commented. This includes header blocks, etc where appropriate. Most lines of the code have comments describing their purpose and is not simply a redefinition of the code.	Most aspects of the code are commented. This includes all header blocks, etc where appropriate. Most lines of the code have comments describing their purpose and is not simply a redefinition of the code.	Most aspects of the code are commented. This includes all header blocks, etc where appropriate. Either little inline comments have been provided, or the comments that are provided are simply restating the code.	The code either has little to no comments or is missing some required information such as header blocks.
Code Structure	The code is written in a way that is very easy to read and follow and follows all conventions taught in class. Only allowed variable types, functions, etc have been used.	The code is written in a way that is easy to read and follow and almost always follows all conventions covered in class. Only allowed variable types, functions, etc have been used.	The way the code is written can be followed but could be improved and almost always follows the conventions covered in class. Only allowed variable types, functions, etc have been used.	The way the code is written can be followed but could be improved. The code generally follows the conventions taught in class. Only allowed variable types, functions, etc have been used.	The code does not follow the conventions taught in class. For example, the variables may be using the wrong naming convention, indentation may not be correct, etc. The code may also be written in a way that is difficult to read and/or follow. And/Or variable types, functions, etc that were not covered in the class material have been used
Program Functionality	The program demonstrates some or all functionality for all parts. All functionality from the previous parts is present. Only allowed variable types, functions, etc have been used.	The program demonstrates some or all functionality for parts 1, 2 and 3. All functionality from the previous parts is present. Only allowed variable types, functions, etc have been used.	The program demonstrates some or all functionality for part 1 and 2. All functionality from the previous parts is present. Only allowed variable types, functions, etc have been used.	The program demonstrates all the functionality of part 1. Only allowed variable types, functions, etc have been used.	The program does not show all the functionality of part 1. And/Or variable types, functions, etc that were not covered in the class material have been used
Programming Techniques used	The program uses all techniques covered in class at least once. These include (but are not limited to): File IO Classes Functions Lists	The program uses all techniques covered in class at least once. These include (but are not limited to): File IO Classes Functions Lists	The program uses most techniques covered in class at least once. These include (but are not limited to): File IO Classes Functions Lists	The program uses some techniques covered in class at least once. These include (but are not limited to): File IO Classes Functions Lists	The program only uses a very limited set of techniques that were covered in class.
Defence of the project	The defence is a hurdle, and you have passed it. All questions asked by the assessors about the submitted code and/or flowchart could be answered to an acceptable level by the student	The defence is a hurdle, and you have passed it. All questions asked by the assessors about the submitted code and/or flowchart could be answered to an acceptable level by the student	The defence is a hurdle, and you have passed it. All questions asked by the assessors about the submitted code and/or flowchart could be answered to an acceptable level by the student	The defence is a hurdle, and you have passed it. All questions asked by the assessors about the submitted code and/or flowchart could be answered to an acceptable level by the student	Questions could not be answered to an acceptable level