

RMIT University

School of Engineering

EEET2248 – Electrical Engineering Analysis

Lectorial Milestone 3

Unit Converter

Lecturer: Dr. Katrina Neville

Student Name: Nathan Paul Williams

Student Number: s3707244

Submission Due Date: 20th April 2018

**Lectorial 2 Milestone**

**Abstract**

This milestone required loops to be added to milestone 2 to allow efficient error checking that looped back to allow the user to retry until valid data was input. It was also required that a user-defined function(s) was implemented that handled the conversions of the user input data to the desired output data.

New tools implemented included the implementation of while loops for error checking. Three while loops are used, with the second and third nested inside the first one. This allows the program to loop back if invalid data is input or if the input is valid the program will simply increment the relevant counter causing that loop to end after the current iteration. User-defined functions are separate lines of code that take in an input(s) and produce an output(s). This code can then be called using the function name and passing in necessary variables. A user-defined function is implemented in our program that takes in the 3 user-input variables (initial system, initial unit and quantity) and outputs 1 double (converted value) , 2 strings (unit names e.g. miles) and 2 error variables that are used to determine if an error has occurred during the function being called. The main program uses these error variables to display relevant error messages and loop back to the correct position after calling the function. The function “isempty()” is used in our user-defined function to check if the quantity variable is empty, if so one of the error variables will increment triggering the error check process. This check is not necessary for the first two user input variables as they are strings, in which case ‘strcmp’ is used to check if they match one of the valid inputs. If not a separate error variable is incremented, triggering another error check process. Inside the user-defined function 'return” is used in two of the ‘error scenarios’ to break the function and return output immediately, this means the function will not finish being called saving run time and also preventing variables being incorrectly modified under a scenario which will cause an error report.

While running this program it appears to have very similar functionality to milestone 2; however it now generates error reports and loops upon invalid input. On the backside however; the programming is much more efficient and concise due to the implementation of the single user-defined function that handles almost all the arithmetic required, keeping the main program much more tidy and readable.

**A screenshot of a cell phone

Description generated with very high confidenceOutput and Testing**

Figure : Standard Output

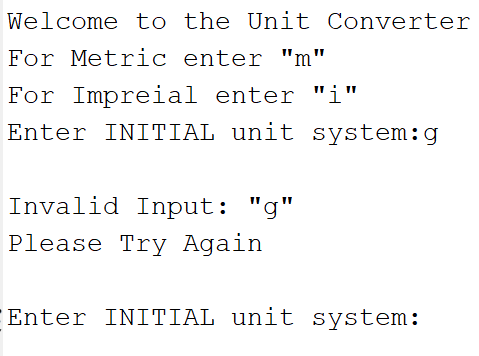


Figure : Error Detection upon entering invalid unit system

A screenshot of a cell phone

Description generated with very high confidence

Figure : Error detection upon invalid string being entered for initial unit type

A screenshot of a social media post

Description generated with very high confidence

Figure : Error detection upon negative number being entered for an invalid unit type

A screenshot of a cell phone

Description generated with high confidence

Figure : Example of how temperature is exempted from the positive input quantity check

A screenshot of a cell phone

Description generated with very high confidence

Figure : Error detection upon blank input for unit quantity

The above figures show the output in different error scenarios. Currently the program handles errors when:

* The input string representing unit system or unit type doesn’t match one of the valid options
* Blank input is entered
* A negative number is entered (except when a temperature is being entered)
* An invalid input (e.g. string) is entered for unit quantity

In all of these scenarios the program will loop back to a point allowing the user to input relevant data again. A focus was also put on usability ensuring the output is displayed nicely in one line in the CLI, the instructions are clearly printed and easy to understand and error messages inform the user of the incorrect input by reprinting the input string/double with a clean error message.

|  |  |
| --- | --- |
| ***Google Unit Converter Output*** | |
| **Original Input** | **Converted Output** |
| 55 degrees Farenheit | 12.7778 degrees Celcius |
| 20 centimetres | 7.87402 inches |
| 5 metres | 16.4042 feet |
| 10 km | 6.21371 miles |
| 40 grams | 1.4096 oz |
| 88 kilograms | 194.007 lbs |
| 100 km/h | 62.1371 mph |
| 20 Litres | 5.28344 |
| 50 hectares | 123.553 |
| -12 degrees celcius | 10.4 degrees Farenheit |
| 5 inches | 12.7 centimetres |
| 20 feet | 6.096 metres |
| 3 miles | 4.82803 km |
| 28 oz | 793.787 grams |
| 200 lbs | 90.7185 kg |
| 88 mph | 141.622 |
| 4 gallons | 15.1416 Litres |
| 75 acres | 30.3514 |

Figure : Output from Google's unit converter

A screenshot of a cell phone

Description generated with very high confidence

Figure 8: Compiled screenshots of program output under identical input

The accuracy tests from last milestone were repeated comparing the programs output to that of Google’s unit converter. The results obtained were identical which was expected considering the formulas used are identical however it was important to ensure no errors had been made when rewriting the code. The comparison once again shows that our output is identical to the google unit converter’s output up except for some which differ slightly after 5 significant figures. It is assumed this is due to slightly different equations being used by google however further testing could be required to ensure that our program is accurate to higher significant places in all scenarios.

Currently our program accurately converts all units from imperial to metric and vice versa, displays this process neatly to the user on the CLI and provides valid error reports and looping upon invalid input. It also uses a user-defined function to handle the arithmetic of the conversion and tidy up the main program increasing readability.