2XC3 Team Assignment

Repository URL

https://github.com/orgs/Sam-Scott-McMaster/teams/team-35-geek-squad/repositories

Title of Project or App

Our tool/App will be called "**Universal Unit Converter**".

High Level Description

A. What kind of app, what will it do, and how will the user use it?

Description

We are going to create a unit conversion app, called "Universal Unit Converter". This app is going to be developed in C. The app will ask the user to enter data in one unit, say Celsius for temperature or meters for distance, and indicate the unit or the quantity they want to convert to, say Fahrenheit or Kilometers. Furthermore, as far as error-handling is concerned, the app will continue to ask the user to input valid numbers if valid numbers are not inputted, ensuring that there are no errors. Moreover, There are several options that will enable extended functionality. One option could be to remove the units in the output. We may add more options depending on the various use cases of the tool.

Usage

Format: convert [-t t|T|d|D|a|A|v|V|m|M] [-i <number><unit>] [-o <unit>] [...options]
Example: convert -t d -i 10m -o km

-t: The type of conversion, t/T represents temperature, d/D represents distance, a/A represents area, v/V represents volume and m/M represents mass.
-i: The initial number that is to be converted. It is formatted as a number, then the shortened unit string. For example, 10m, 32ft, 1C, etc…
-o: The unit of the desired output unit. This is also in the shortened unit string. For example, 31mm, 54yd, 3L.
(option) -ru: This option removes the unit from the output. So if you want to convert 1000m to km, it would result in an output of 1 instead of 1km.

B. What is the context in which this app will be used?

The purpose of this application is towards anyone who wants to convert between units in some sort of work such as (but not limited to); research, creation, science, etc.

C. What constraints are there on your design?

Programming Language

The app is built in C, which limits certain high-level functionality and requires careful memory and resource management.

Data Handling

The app needs to handle and accurately convert between units with minimal floating-point errors, handling a lot of data will make it very tedious for us. To reduce this workload, we plan on using the maps data structure to convert units from one to another.

Memory Allocation/Limits:

The app can only support up to a certain amount of data. For instance, the user may not be able to convert very large numbers outside C's memory limit. To combat this, we can store prefix multiplier values as strings within a dictionary. When needed the prefix can be converted to an integer allowing just that space. The largest number the app will be able to handle is 9,999,999,999,999 and the smallest is -9,999,999,999,999 (less than +quadrillion, greater than -quadrillion).

Algorithm clutter:

conversion from metric to metric units is fairly simple as prefixes follow an exponential pattern with base 10. However, more complex functions are needed to complete the calculations with imperial-to-imperial calculations or metric-to-imperial calculations.

Supported Conversions

The following list demonstrates the types of measurements that are supported using this tool initially. Throughout the design process, this may change.

1. Temperature
    - Kelvin, Celsius, and Fahrenheit.
2. Distance
    - Imperial: Inch, **Foot**, Yard, Mile.
    - Metric: Terameter, Gigameter, Megameter, kilometer,..., **Meter**, …, Millimeter, Micrometer, Nanometer, Picometer.

3. Area
    - Imperial: Inch^2, **Foot^2**, Yard^2, Mile^2
    - Metric:Terameter^2, Gigameter^2, Megameter^2, Kilometer^2,..., **Meter^2**, …, Millimeter^2, Micrometer^2, Nanometer^2, Picometer^2
4. Volume
    - Imperial: Inch^3, **Foot^3**, Yard^3, Mile^3
    - Metric:Terameter^3, Gigameter^3, Megameter^3, Kilometer^3,..., **Meter^3**, …, Millimeter^3, Micrometer^3, Nanometer^3, Picometer^3
5. Mass
    - Imperial: Ounce, Pound, Stone
    - Metric: Teragram, Gigagram, Megagram, Kilogram,..., **Gram**, …, Milligram, Microgram, Nanogram, Picogram.

****NOTE**** The unit that is bolded is considered the “base” unit for that unit system, meaning that any unit that is not a base, first gets converted to the base, then to the desired unit. This is done to reduce memory used for the conversions.

Team Members

Name: Arian Fallahpour-Sichani
Student ID: 400463417
Github ID: arian-fallahpour
Role Titles: Frontend

Name: Soham Hajariwala
Student ID: 400516310
Github ID: Soham1-coder
Role Titles: Converter

Name: Naqeeb Ahmadzai
Student ID: 400505872
Github ID: Ahmadzan01
Role Titles: Error Handling

Name: Krish Haryani
Student ID: 400523256
Github ID: krishharyani
Role Titles: Backend

Describing Your Role

Arian Fallahpour-Sichani - Develops the user interface, ensures members do tasks.

Soham Hajariwala - Converter, will handle the mathematical operations behind the conversion process.

Naqeeb Ahmadzai - Error handling, will handle all the incorrect/invalid inputs from the user, this will include invalid conversions (i.e converting from temperature to distance).

Krish Aryani - Will handle the algorithms behind the conversions and expand on different conversion types

Increments

- (At least 1) What functionality from each team member will be in each increment, what is your target data for each increment?

  Get the input from the user (front end)
- Grep commands to extract input
- Target Date: November 13

Figure out conversion algorithms for a small selection of units

- Some starting units may include:
- Temperature: Kelvin to celcius
- Distance: Standard metric to imperial units
- Target Date: November 13

Error Handling

- Type errors (incompatible unit conversion)
- Invalid input handling
- Memory errors
- Target Date: November 17

Build upon current unit conversion algorithms to support more units

- More complex may include: Area, Volume, Mass
- Target Date: November 18