Assignment 2 - Manipulating Numeric Data

Assigned: Wednesday, September 9
Due: Thursday September 24, *before midnight*
Value: 20 points (for successfully submitting a functionally correct program to your Assembla repo before the deadline). Late submissions will receive a 0.

Executive Summary

You will develop a program that performs various conversions (time, weight, temperature, currency, distance).

Purpose

- Learn how to input and output numeric values using formatted I/O library functions
- Learn how to define and symbolically manipulate numeric data
- Learn how to use basic programming constructs, specifically loops and conditionals
- Sharpen your problem solving skills and general programming skills

Deliverables

The `assignment-2.c` program source file implementing the functionality described below. All your code should go into this file. We will grade the latest version of your code pushed to your Assembla repo before the due date. **BE SURE TO COMMIT YOUR CODE TO SVN!**

In order to be graded for your work, your `assignment-2.c` **MUST** #include the provided `assignment-2.h` file and implement all functions within. Do not edit `assignment-2.h`! Any edits you make would not be reflected during grading - we will use only your `assignment-2.c` file with `assignment-2.h` identical to what we provide you.

Introduction

Our customer is planning a trip to Ireland, and would like a program that can help him perform conversions between the following:

- Irish time and Austin time
- Euros and US Dollars
- Fahrenheit and Celsius
- Kilograms and Pounds
- Kilometers and Miles

In each case, you will have to support conversions in both directions.

Time

Your input will be two whole numbers, the hour (which ranges from 0 to 23, inclusive) and the number of minutes (which ranges from 0 to 59, inclusive). For example, when Austin is on CST and it is 9:00am in Austin, it is 3:00pm in Ireland, on the same day. All times will be in 24 hour format, in which Ireland time = (Austin hours + 6) mod 24 and Ireland minutes = Austin minutes.

Currency

Ireland uses the Euro. To convert from Euros to US Dollars, multiply the quantity of Euros by **1.36.**

When converting from US Dollars to Euros, the input will be given as two integers (the dollar quantity and the cents quantity). Your output should be the Euro equivalent expressed as a real number. The conversion rate is **.74**

When converting from Euros to US Dollars, the input will be given as a real number. Your output should be the converted number of dollars and cents, i.e., reported as two whole numbers. The number of cents should be rounded to the nearest whole number.

Temperature

The relationship between temperature readings in Fahrenheit and in Celsius is given by the following equations:

```
C = (5/9)(F-32)
F = (9/5)C + 32
```

(This is not valid C syntax!)

Fahrenheit temperatures must be expressed in integers. Celsius temperatures are real numbers.

Your program must convert a given Fahrenheit temperature, expressed as an integer, to Celsius, expressed as a real number, and vice versa. When converting from Celsius to Fahrenheit use rounding to get an integer.

Weight

The kilogram is the basic unit of weight used in Ireland. One kilo equals 1000 grams. The basic unit of weight in the US is the pound. One pound equals 16 ounces.

One pound equals exactly 453.59237 grams.

US weights will be expressed as a pair of whole numbers denoting the number of pounds and ounces. Kilograms will be expressed as real numbers.

When converting from pounds and ounces into Kilograms, the result is a real number representing the equivalent number of kilograms. When converting from kilograms to pounds and ounces, round to the nearest whole number.

Distance

One kilometer equals 0.6213712 miles. All distances will be expressed as real numbers.

Requirements

Setup

1. Create a Project named *Lab2* in Visual Studio for this lab. Be sure to uncheck the "Create directory for solution" option. The project location should be your <repo> folder.
   In the project wizard, make sure to check "Empty project" and uncheck "Security Development Lifecycle (SDL) checks"
2. From the Solution Explorer, add the file `assignment-2.c` to your *Lab2* project, so your file structure will be <repo>/`Lab2/assignment-2.c`. At the top of the file, be sure to include the header comment as described in the syllabus.
3. Copy the file `assignment-2.h` to the same folder where `assignment-2.c` is. Then, from the Solution Explorer, add the `assignment-2.h` file to your project. (Add -> Existing Item)

Input format

Your program should prompt for and then receive input value for the selected conversion to be performed and also the values to be converted. All input is read from the keyboard using the *scanf* library function. Output messages (described later) prompt the user for inputs.

Your program should work correctly for all legitimate values that are input.

If a bad input value is given, then your program may terminate or give a bad output value.

When you need to prompt for multiple input values, ask for the input values in the following orders:

- When prompting for time, let the first number be the hour and the second number be minutes.
- When prompting for money, let the first number dollars and the second number be cents.
- When prompting for weight, let the first number be pounds and the second number be ounces.

Output format

At the start of conversions, output the menu below, followed by the prompt for a conversion.

Use the *printf* library function for all output

Conversion menu

1. Convert a given Austin time to Irish time
2. Convert a given Irish time to Austin time
3. Convert a given USD value to EUR
4. Convert a given EUR value to USD value
5. Convert a given Fahrenheit temperature to Celsius
6. Convert a given Celsius temperature to Fahrenheit
7. Convert a given weight in kg to pounds, ounces

8. Convert a given weight in pounds, ounces to kg
9. Convert a given distance in km to miles
10. Convert a given distance in miles to km
11. Stop doing conversions and quit the program

The results of each conversion should be output to the terminal window, following the output format specified below for each conversion. Values below in <brackets> should be replaced by the appropriate value when being output. An example input/output dialog follows after all output formats.

1. Austin time to Irish time

```
The time in Ireland equivalent to <hr>:<min> in Austin is <hr>:<min>
of the <previous/same/next> day.
```

2. Irish time to Austin time

```
The time in Austin equivalent to <hr>:<min> in Ireland is <hr>:<min>
of the <previous/same/next> day.
```

3. USD to EUR

```
<Euro value> Euros is equivalent to $<dollars>.<cents> in US
Dollars.
```

4. EUR to USD

```
$<dollars>.<cents> in US Dollars is equivalent to <Euro value>
Euros.
```

5. Fahrenheit to Celsius

```
<temp in C> Celsius is equivalent to <temp in F> Fahrenheit.
```

6. Celsius to Fahrenheit

```
<temp in F> Fahrenheit is equivalent to <temp in C> Celsius.
```

7. Kilograms to Pounds, Ounces

```
<pound component of weight> lb <ounce component of weight> oz is
equivalent to <weight in kilograms> kg.
```

8. Pounds, Ounces to Kilograms

```
<weight in kilograms> kg is equivalent to <pound component of
weight> lb <ounce component of weight> oz.
```

9. Kilometers to Miles

```
<distance in miles> mi is equivalent to <distance in kilometers> km.
```

10. Miles to Kilometers

`<distance in kilometers> km is equivalent to <distance in miles> mi.`

An example of the dialog between the user and your program is given below (where `<conversion menu>` should be replaced with a full output of the menu), with user input being shown in boldface:

```
<conversion menu>
Enter a number from the menu (1-11) to select a specific
conversion to perform or to quit: 2
Enter an Irish time to be converted, expressed in hours and
minutes: 1 30
The time in Austin equivalent to 1:30 in Ireland is 19:30 of
the previous day.
<conversion menu>
Enter a number from the menu (1-11) to select a specific
conversion to perform or to quit: 11
Good Bye
```

Design and implementation constraints

- Use the formatted IO functions presented in class and found in your textbook.
    - Input specific values from the standard input device, i.e., keyboard, using *scanf()*
    - Use the *printf()* function as shown in class to output the menu, the prompts, and your program's results to the standard output device, i.e., monitor/terminal window.
- Use symbolic formulae for each specific conversion.
- Use appropriately named variables to represent the given and converted values in each category.
- Use the *int* and *double* data types for the variables needed by each conversion formula.
- Use `"%lf"` for *scanf()* for reading in `double`
- Use meaningful variable names so that your program is understandable by other humans, in addition to being legal identifiers according to the C compiler.
- Use **const** identifiers in upper case (CAPS) for the conversion factors.

.

Lab Specific FAQ

- What does it mean to round a real number to the nearest whole number?

  Every nonnegative real number is sandwiched uniquely between two successive whole numbers, e.g., 3.14 lies in [3,4]. Rounding entails returning the whole number that closest to the given real number (so 3.14 rounds to 3)

- How do I round a real number that is exactly halfway between two successive whole numbers?

Round up, i.e., 42.5 cents should be rounded to 43 cents, 3.5 ounces should be rounded to 4 ounces, etc.

- How do I get a double with scanf function?

  Use “%lf” as you get a compiler error or absurd values when you use “%f”

- How many significant places after the decimal point should we have?
  Its 6 by default

- Do we have to account for invalid input such as characters or numbers outside the expected range?

  No. You do not need to handle invalid input or user error as we will grade how your program functions with valid input as specified in the lab document.