C++ Mickey's Multiple Unit Converter

Contents

Time required: 90 minutes

Formatting Numbers in C++

When you output a double by using cout, the resulting decimal places are what they are. You have no control over it.

```
double division = 4.0 / 3.0;
std::cout << "4.0/3.0 = " << division << std::endl;
```

The result is as follows:

```
4.0/3.0 = 1.33333
```

C++ provides a couple of ways of setting precision. A method called **printf** gives you more control of the format. The "f" in printf stands for "formatted". Here's an example:

```
printf("4.0/3.0 = %.2f", division);
```

The first value in the parentheses is a format string that specifies how the output should be displayed. This format string contains ordinary text followed by a format specifier, which is a special sequence that starts with a percent sign. The format specifier %.3f indicates that the following value should be displayed as floating-point, rounded to three decimal places:

```
4.0/3.0 = 1.33333
4.0/3.0 = 1.33
```

The format string can contain any number of format specifiers; here's an example with two of them:

```
double CM_PER_INCH{2.54};
int inch = 100;
double cm = inch * CM_PER_INCH;
printf("\n%d in = %.3f cm\n", inch, cm);
```

The result is as follows:

```
100 in = 254.000 cm
```

printf does not append a newline. **\n** is needed to add a newline character.

The format specifier %d displays integer values ("d" stands for "decimal", meaning base 10 integer). The values are matched up with the format specifiers in order, so inch is displayed using %d, and cm is displayed using %f.

Learning about format strings is like learning a sublanguage within Java. There are many options, and the details can be overwhelming. The table below lists a few common uses, to give you an idea of how things work.

%d	Integer in base 10 ("decimal")	12345
%,d	Integer with comma separators	12,345
%08d	Padded with zeros, at least 8 digits wide	00012345
%f	Floating-point number	6.789000
%.2f	Rounded to 2 decimal places	6.79
%s	String of characters	"Hello"
%x	Integer in base 16 ("hexadecimal")	bc614e

Assignment: Unit Converter

This program will start by asking the user what type of conversion they wish.

Create a program named **UnitConverter.cpp**

From a menu, the user can choose from the following conversions. Each conversion will have a separate function.

- Cm to Inches

- Inches to Cm
- Km to Miles
- Miles to Km

Requirements

- Round the results to 2 decimal places.
- You will want to use a while loop for the menu.
- Input Validation: We want the input to be in the right range. If the user enters a negative length, the program should tell the user that the entry is invalid.
- Otherwise, the program should convert the length and print out the result.

Example run:

```
 +-------------------------------------------------+
 |             Multiple Unit Converter             |
 |     Miles, Kilometers, Inches, Centimeters      |
 +-------------------------------------------------+

[1] Kilometers to Miles
[2] Miles to Kilometers
[3] Centimeters to Inches
[4] Inches to Centimeters
[9] Exit
Enter menu selection: 1
Enter kilometers: 23
23.00 Kilometers is 14.29 Miles
[1] Kilometers to Miles
[2] Miles to Kilometers
[3] Centimeters to Inches
[4] Inches to Centimeters
[9] Exit
Enter menu selection: 2
Enter miles: 34
34.00 Miles is 54.72 Kilometers
[1] Kilometers to Miles
[2] Miles to Kilometers
[3] Centimeters to Inches
[4] Inches to Centimeters
[9] Exit
Enter menu selection: 3
Enter centimeters: 43
43.00 Centimeters is 16.93 Inches
[1] Kilometers to Miles
[2] Miles to Kilometers
[3] Centimeters to Inches
[4] Inches to Centimeters
[9] Exit
Enter menu selection: 4
Enter inches: 1
1.00 Inches is 2.54 Centimeters
[1] Kilometers to Miles
[2] Miles to Kilometers
[3] Centimeters to Inches
[4] Inches to Centimeters
[9] Exit
Enter menu selection: 9
Quitting ...
```

Assignment Submission

1. Attach the pseudocode.
2. Attach the program files.
3. Attach screenshots showing the successful operation of the program.
4. Submit in Blackboard.