Assignments

Assignment 2 - Returned

Honor Pledge Accepted
Draft - In progress
Submitted
Returned

Assignment Details

Title	Assignment 2
Student	Ramtin Sirjani
Submitted Date	Oct 10, 2022 8:31 PM
Grade	**100.00 (max 100.00)**
History	Oct 10, 2022 8:31 PM EDT Ramtin Sirjani (rsirjani) saved draft Oct 10, 2022 8:31 PM EDT Ramtin Sirjani (rsirjani) submitted

Instructions

Assignment overview

We would like students to experience command line input with C types of character, int, and float, to understand and use C types such as char, int, and float, as well as the flow control structures studied in class.

This assignment consists of two parts.
In part one, you are required to write a C program to perform some simple conversions.
In part two, you are to write a C program to translate integers into the equivalent English word.

Part one: 70%

The goal of the exercise is to implement a simple converter, called ”converter”, which works as follows.

1. First, the user is asked what she/he wants to do. An integer can be entered with the following six actions associated with different values of the integer. You can assume that the user will always enter an integer.
   - 1 for conversion between Kilograms and Pounds (1 kilogram == 2.20462 pounds)
   - 2 for conversion between Hectares and Acres (1 hectare == 2.47105 acres)
   - 3 for conversion between Litres and Gallons (1 litre == 0.264172 gallons)
   - 4 for conversion between Kilometre and Mile (1 kilometre == 0.621371 miles)
   - 5 for quit
   - {any other integer} for prompting the user to try again.
2. In case of 1 to 4, the program will ask the direction of the conversion. In each case, one of two characters can be entered corresponding to each conversion direction. Your program should handle non-valid single character input values.
   - In case of 1, the program will prompt the user for two choices and wait for a character input
     - K for conversion from Kilograms to Pounds
     - P for conversion from Pounds to Kilograms
   - In case of 2, the program will prompt the user for two choices and wait for a character input
     - H for conversion from Hectares to Acres

        - A for conversion from Acres to Hectares
    - In case of 3, the program will prompt the user for two choices and wait for a character input
        - L for conversion from Litres to Gallons
        - G for conversion from Gallons to Litres
    - In case of 4, the program will prompt the user for two choices and wait for a character input
        - K for conversion from Kilometre to Mile
        - M for conversion from Mile to Kilometre
    - Note/Hint: Since whitespace counts as a character, to read a character properly, your program should handle the leading space character, tab character, and end of line character, if any.
3. Then the program descriptively prompts the user for the input value ("Please enter a value..."), descriptively displays the result ("Your conversion is..."), and returns to the main menu. The input value should be a float number and we assume the user will always enter valid numbers.
4. Your program should follow good programming styles, i.e. write clear code, choose good variable names, make proper comments, etc. Your program must have a comment block at the top describing the name of the program, it's purpose, it's author (ie. you and your student number), etc.

Part two: 30%

1. Write a succinct program to convert any integer from 1-999 into the English word (in lower case). Eg.

    -
      ```
      Please enter a value (1-999, 0 to quit): 1
      You entered the number one

      Please enter a value (1-999, 0 to quit): 15
      You entered the number fifteen

      Please enter a value (1-999, 0 to quit): 47
      You entered the number forty-seven

      Please enter a value (1-999, 0 to quit): 135
      You entered the number one hundred and thirty-five

      Please enter a value (0 to quit): 0
      ```
2. You can assume the input is always one valid integer in the range 0-999 (ie. no negatives and no integers higher than 999). The integer will never have preceding zeros (eg. The input would be 1, never 001) or trailing decimal zeros (eg. The input would be 1, never 1.000).
3. Then the program descriptively prompts the user for the input value ("Please enter a value..."), descriptively displays the result ("You entered the number..."), and returns to the main menu.
4. Your program should follow good programming styles, i.e. write clear code, choose good variable names, make proper comments, etc. Your program must have a comment block at the top describing the name of the program, it's purpose, it's author (ie. you and your student number), etc.

Testing and submitting

You should test your program by compiling and running it on Gaul before submitting (This is how the TA will be testing your program). For part one, each case should be tested at least once. For part two, you should test at least two integers 1-9, at least two integers 10-19, at least two integers 20-99, and at least two integers 100-

999. Capture your testing by submitting a screenshot. There should be two screenshots. One for each part. (You may submit extra images if you cannot fit all your testing in one screenshot)

You will then need to:

- Remove any compiled programs and other files you may have. You should have exactly two C programs in a directory called 251xxxxxx-Assignment2 where 251xxxxxx is your student number.
- Upload your screenshots to Gaul. Place them in your Assignment 2 directory where your C programs are located. Your directory structure should look like this:

```
251xxxxxx-Assignment2
   |----- converter.c
   |----- converter.png
   |----- intToEnglish.c
   |----- intToEnglish.png
```

- Create a tarball out of your directory called 251xxxxxx-Assignment2.tar
- Download the tarball from Gaul to your local workstation
- Submit the single tarball called 251xxxxxx-Assignment2.tar in OWL

I need help

Check to see if your question has already been posted in the forums. If not, post your question.

https://owl.uwo.ca/x/yvBGmc

Submitted Attachments

- 250680632-Assignment2.tar (280 KB; Oct 10, 2022 8:31 pm)

Assignment 2 - C basics

Program 1 - Organization and Style

Instructions followed. Properly documented and commented. Useful naming conventions. Appropriate spacing.

Inadequate or Incomplete

Did not follow instructions at all. Commenting is inadequate or missing. Naming is consistently inadequate. Spacing and program layout is difficult to follow.

0 Points

Poor

Did not follow many key instructions. Commenting is generally inadequate. Naming is generally inadequate. Spacing and program layout is difficult to follow.

3.5 Points

Fair

Followed most instructions. Commenting is generally okay. Naming is generally okay. Spacing and program layout is generally okay.

5 Points

Good

Followed most instructions. Commenting is generally good. Naming is generally good. Spacing and program layout is generally good.

7.5 Points

Exceptional

Followed instructions. Commenting is exceptional. Naming is exceptional. Spacing and program layout is exceptional.

10 Points

Comment for Program 1 - Organization and Style

Done

0 0

Program 1 - Functionality

Program compiles. Program does what is asked. Program is not buggy.

Inadequate or Incomplete

Program does not compile or compiles with many adjustments. Program is very buggy and/or rarely produces correct results. Or, no submission received.

0 Points

Poor

Program does compile or compiles with minimal adjustments. Program is very buggy and/or rarely produces correct results.

15 Points

Fair

Program does compile. Program has only few bugs and usually produces correct results.

30 Points

Good

Program does compile. Program has no obvious bugs and usually produces correct results.

45 Points

Exceptional

Program does compile. Program has no bugs and consistently produces correct results.

60 Points

Comment for Program 1 - Functionality

Done

0 0

Program 2 - Organization and Style

Instructions followed. Properly documented and commented. Useful naming conventions. Appropriate spacing.

Inadequate or Incomplete

Did not follow instructions at all. Commenting is inadequate or missing. Naming is consistently inadequate. Spacing and program layout is difficult to follow.

0 Points

Poor

Did not follow many key instructions. Commenting is generally inadequate. Naming is generally inadequate. Spacing and program layout is difficult to follow.

1.25 Points

Fair

Followed most instructions. Commenting is generally okay. Naming is generally okay. Spacing and program layout is generally okay.

2.5 Points

Good

Followed most instructions. Commenting is generally good. Naming is generally good. Spacing and program layout is generally good.

3.75 Points

Exceptional

Followed instructions. Commenting is exceptional. Naming is exceptional. Spacing and program layout is exceptional.

5 Points

Comment for Program 2 - Organization and Style

Done

0 0

Program 2 - Functionality

Program compiles. Program does what is asked. Program is not buggy.

Inadequate or Incomplete

Program does not compile or compiles with many adjustments. Program is very buggy and/or rarely produces correct results. Or, no submission received.

0 Points

Poor

Program does compile or compiles with minimal adjustments. Program is very buggy and/or rarely produces correct results.

6.25 Points

Fair

Program does compile. Program has only few bugs and usually produces correct results.

12.5 Points

Good

Program does compile. Program has no obvious bugs and usually produces correct results.

18.75 Points

Exceptional

Program does compile. Program has no bugs and consistently produces correct results.

25 Points

Comment for Program 2 - Functionality

Done

0 0

Total: **0**

Back to list