Chapter 2 Program 2 Documentation

Ashton Hellwig

February 16, 2020

Contents

List of Algorithms

Listings

1 Problem Analysis

The problem states:

> This assignment relates to content from Chapter 2 of the eText.
>
> **Instructions**
>
> 1. Review the general programming assignment instructions.
> 2. Write a program that prompts the capacity, in gallons, of an automobile fuel tank and the miles per gallon the automobile can be driven. The program outputs the number of miles the automobile can be driven without refueling. Numbers entered for capacity must allow entry of capacity being an integer and the miles per gallon in decimals. The number of miles must be output to the next lowest integer (without decimals).

1.1 Data

No given data points. Available data types include:

1. *fuelCapacity*: Must be an `int`.
2. *milesPerGallon*: Must be a decimal value (`float` or `double`).
3. *fuelRange*: Must be the next lowest `int` without decimals.

1.2 Desired Output

```
1 Please enter your vehicle's fuel capacity (must be a whole number integer):
2 Please enter your vehicle's gas milage (decimal-point numbers allowed):
3
4 You are able to drive |fuelRange| miles without refueling
```

Listing 1: main.cpp output

2 Algorithm

Below is the algorithm for the program.

Algorithm 2.1 Chapter 2 Program Algorithm

```
1:  function CALCFUELRANGE(fuelCapacity, gasMilage)
2:      fuelRange ← 0                                   ▷ Declare varible for return value
3:
4:      fuelRange ←CASTTOINT(fuelCapacity ∗ gasMilage)
5:      return fuelRange
6:  end function
7:
8:  function MAIN                                       ▷ –Variable Declarations–
9:      fuelCapacity ← 0
10:     gasMilage ← 0
11:     fuelRange ← 0
12:                                                     ▷ Prompt user for values
13:     TOOUTPUT ( " Please enter your vehicle's fuel capacity (must be a whole number integer): " )
14:     fuelCapacity ← USERINPUT
15:
16:     TOOUTPUT (" Please enter your vehicle's gas milage (decimal-point numbers allowed): ")
17:     gasMilage ← USERINPUT
18:
19:     fuelRange ← CALCFUELRANGE(fuelCapacity, gasMilage)   ▷ Calculate fuel range
20:                                                     ▷ Output fuel range using above function
21:     TOOUTPUT ("You can drive [fuelRange] miles without refueling.")
22: end function
```

3 User Documentation

3.1 Build

The following are instructions with two use cases:

- With CMake
- Bundled Release

3.1.1 With CMake

1. Navigate to the unzipped folder containing the binary, **with a terminal emulator or command prompt**, this will (most likely) mean running:

```
cd ~/Downloads/ashellwig_csc160_programming-assignment_m1c2p2
```

2. Compile the program using CMake after switching to the build directory:

```
cd build
cmake \
  -G "Unix Makefiles" \
  -DCMAKE_CXX_COMPILER=clang++ \
  -DCMAKE_CXX_FLAGS="-Wall -std=c++2a -g -glldb" ..
cmake --build .
./Chapter2Program2  # Run the compiled program right away
cd ..  # Return to project directory
```

3.1.2 Bundled Release

1. Navigate to the unzipped folder containing the binary, **with a terminal emulator or command prompt**, this will (most likely) mean running:

```
cd ~/Downloads/ashellwig_csc160_programming-assignment_m1c2p2
```

2. To run the program simply issue this within the command prompt

```
./build/Chapter2Program2
```

Of course if preferred, you may also navigate to the build folder in file explorer and double click the executable (`Chapter2Program2`).

4 Images

```
↵ 1  ahellwig@archwig  ~/Documents/School/CSC160/Module-1/Programming-Assignments/Chapter2Program2/build
cmake -G "Unix Makefiles" -DCMAKE_CXX_COMPILER="/usr/sbin/clang++" -DCMAKE_CXX_FLAGS="-Wall -std=c++2a -g -gllo

-- The C compiler identification is GNU 9.2.1
-- The CXX compiler identification is Clang 9.0.1
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/sbin/clang++
-- Check for working CXX compiler: /usr/sbin/clang++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/ahellwig/Documents/School/CSC160/Module-1/Programming-Assignments/Ch
 ✔ 〉ahellwig@archwig  ~/Documents/School/CSC160/Module-1/Programming-Assignments/Chapter2Program2/build
make[1]: Entering directory '/home/ahellwig/Documents/School/CSC160/Module-1/Programming-Assignments/Chapter2Pr
make[2]: Entering directory '/home/ahellwig/Documents/School/CSC160/Module-1/Programming-Assignments/Chapter2Pr
Scanning dependencies of target Chapter2Program2
make[2]: Leaving directory '/home/ahellwig/Documents/School/CSC160/Module-1/Programming-Assignments/Chapter2Pro
make[2]: Entering directory '/home/ahellwig/Documents/School/CSC160/Module-1/Programming-Assignments/Chapter2Pr
[ 25%] Building CXX object CMakeFiles/Chapter2Program2.dir/src/general_functions.cxx.o
[ 50%] Building CXX object CMakeFiles/Chapter2Program2.dir/src/chapter2.cxx.o
[ 75%] Building CXX object CMakeFiles/Chapter2Program2.dir/src/main.cxx.o
[100%] Linking CXX executable Chapter2Program2
make[2]: Leaving directory '/home/ahellwig/Documents/School/CSC160/Module-1/Programming-Assignments/Chapter2Pro
[100%] Built target Chapter2Program2
make[1]: Leaving directory '/home/ahellwig/Documents/School/CSC160/Module-1/Programming-Assignments/Chapter2Pro
 ✔ 〉ahellwig@archwig  ~/Documents/School/CSC160/Module-1/Programming-Assignments/Chapter2Program2/build
Please enter your vehicle's fuel capacity (must be a whole number integer): 10
Please enter your vehicle's gas milage in miler per gallon (decimal-point numbers allowed): 12.32
You are able to drive approximately 123 miles without refueling.
Press enter to continue...%
```