C FUNDAMENTALS PRACTICE - VARIABLES

Table of Contents

1. READ README

- This file is a practice file for C fundamentals - based on the lecture `4_variables.org` (GitHub). You find this file at: tinyurl.com/4-variables-practice-org.
- When you're done with a section move the cursor on the section heading and type `S-<right>` (or SHIFT+ <right-arrow>).
- When you leave class without having completed the file, save a copy to GDrive as a backup and/or to work on it from home
- When you've completed the file, upload it to Canvas

2. DONE Identify yourself

- Replace `[yourName]` in the header of this file by your name
- Add `(pledged)` next to your name (as in "I obey the honor code")
- Change the "TODO" in the headline to "DONE" (`S-<right>`)
- Save the file (`C-x C-s`).

3. DONE Variable types and declarations

1. Create a named C code block named 1 below.
2. Declare two *floating-point* variables `fahrenheit` and `celsius`.
3. Use two separate statements: one declaration per line.
4. Run the code block (`C-c C-c`).

—— PUT CODE BLOCK BELOW THIS LINE ——

```
float fahrenheit;
float celsius;
```

4. DONE Fix the program

1. A couple of things are wrong in the code block 1.

2. You can check that yourself by running it (`C-c C-c`) and reading the compiler messages that open in another buffer. Type `C-x 1` to only see 1 buffer.
3. Find and fix the errors, and run the code block to make sure. Notice how the syntax highlighting changes!

```
float freezing_point;
freezing_point = 32.0f;
```

Second solution:

```
float freezing_point = 32.0f;
```

5. DONE Variable assignments

1. Create a code block 1 below.
2. Declare **and** initialize two *floating-point* variables, `freezing` and `factor`, with the values `32` and `5/9`, respectively.
3. Declare and initialize these variables in **one** statement only.

—— PUT CODE BLOCK BELOW THIS LINE ——

```
float freezing = 32.f, factor = 5.f/9.f;
```

6. DONE Variable computations

1. The code from 1 and from 1 has been copied into the code block 1 below[1].
2. Complete 1 with two statements:
   - assign the temperature 80 to `fahrenheit`
   - compute `celsius` using 1 (which will not run - why?)

     ```
     celsius = (fahrenheit - freezing) * factor;
     ```

3. Run the program to make sure that the answer is correct for 80 degrees Fahrenheit (equivalent to 26.7 degrees Celsius).

```
float fahrenheit;
float celsius;  // declares fahrenheit and celsius

float freezing = 32.f, factor = 5.f/9.f;
 // assigns values to freezing and factor
fahrenheit = 32.0f; // assign 80 F to the variable fahrenheit
celsius = (fahrenheit - freezing) * factor; // add the pgm:formula codeblock here
printf("Fahrenheit: %g\nCelsius equivalent: %.1f\n",
       fahrenheit, celsius);
```

```
Fahrenheit: 32
Celsius equivalent: 0.0
```

7. DONE Fix the program

1. The program 1 declares and initializes the variable `i` with the value `0`. After assigning `1` to `i`, it should print out `1` but it prints `0` instead.
2. Fix the error and then run the block with `C-c C-c` to check.

```
int i = 0;
i = 1;
printf("%d\n", i);
```

```
1
```

8. DONE Formatting printout

1. Define and initialize three variables in a code block named 1:
   - an integer variable `foo` with value 100
   - a floating-point variable `bar` with value 100
   - a character variable `baz` with value A
2. Print the three variables so that the output looks like shown below. Remember to add `:results output` to the code block header.
3. Use
   - `puts` for the headline "Three variables",
   - `printf` to print `foo` and `bar`, and
   - `putchar` to print `baz`.

   Tip: The final program 1 has at least 7 lines.

   Output:

   ```
   Three variables:
   foo = 100
   bar = 100.01
   baz = A
   ```

—— PUT CODE BLOCK BELOW THIS LINE ——

```
// declarations
int foo;
float bar;
char baz;

// assignments
foo = 100;
bar = 100.01f;
baz = 'A';

// print output
puts("Three variables:");
printf("foo = %d\n", foo);
printf("bar = %.2f\n", bar);
```

```
printf("baz = ");
putchar(baz);
```

```
Three variables:
foo = 100
bar = 100.01
baz = A
```

```
int foo = 100;
float bar = 100.01f;
char baz = 'A';

puts("Three variables:");
printf("foo = %d\nbar = %.2f\nbaz = ", foo, bar);
putchar(baz);
```

```
Three variables:
foo = 100
bar = 100.01
baz = A
```

9. DONE Fix the program

The program [1] should print out

```
Speed of light (m/s): c = 299792458
Euler number: e = 2.7183
```

But instead it prints out this:

```
Speed of light (m/s): c = 14.985029
Euler number: e = 0
```

Fix the program to get the right output!

```
int c = 299792458;
float e = 2.718282f;

printf("Speed of light (m/s): c = %d\n", c);
printf("Euler number: e = %.4f\n", e);
```

```
Speed of light (m/s): c = 299792458
Euler number: e = 2.7183
```

Footnotes:

[1] The header argument `:noweb` enables referencing to other code. Setting it to `yes` means that references are expanded when evaluating, tangling, or exporting. You can check that by tangling the source code and looking at

the result (more info).

Author: Marcus Birkenkrahe (pledged)

Created: 2024-02-16 Fri 23:51

Validate