Variables and Assignments

See also King chapter 2 - for CSC 100 Intro to Programming in C/C++

README

- This file accompanies the lecture on variables and assignments in C. To gain practice, you should type along in your own Org-mode file.
- To make this easier, use the auto expansion (`<s`), add the following two lines at the top of your file, and activate each line with `C-c C-c` (you should get a confirmation in the minibuffer):

```
#+PROPERTY: header-args:C :main yes
#+PROPERTY: header-args:C :includes <stdio.h>
```

- Remember that `C-M-\` inside a code block indents syntactically

Variable types and declarations

- C computes using placeholders, or **variables**
- Each variable must have a **type** to specify the data it can hold
- E.g. `int` (integer), `float` (floating point), `char` (character)
- Variables must be **declared** before they can be used, see 1

```
int height;
float profit;
char name;
```

- Several variables of the same type can be declared together:

```
int height, length, width, volume;
float profit, loss;
char first_name, last_name;
```

- Variable type declarations must precede statements that use the variables
- The block with **declarations** comes before the statements[1]

Variable assignment

- A variable gets its value through **assignment**
- In 1, the variable `height` gets the value `8`[2]

```
height = 8;
```

- [X] If you tried to run 1, you got an error. Can you see why?[3]
- [X]

  However, 1 throws another error. What's wrong this time?[4]

```
height = 8;
int height;
```

- Code block 1 works.

```
int height;
height = 8;
```

- A constant assigned to a `float` variable contains a decimal point and the letter `f`, as shown in 1.

```
float profit;
profit = 2150.48f;
```

- Assigning a `float` to an `int` and vice versa is possible but not safe.
- Variables with values can be used to compute other values, as shown in 1.

```
int height, length, width, volume;

height = 8;
length = 12;
width =10;
volume = height * length * width;
```

- To print these variables, we need to learn **formatting**

Formatting printout

- We use the built-in (via `stdio.h`) function `printf` to print
- In the code 1, `%d` is a placeholder for an `int`:

```
int height;  // type declaration
height = 8;  // variable assignment

printf("The height is: %d\n", height); // formatted printout
```

```
The height is: 8
```

- In 1, `%f` is used to print a `float`:

```
float profit;       // type declaration
profit = 2150.48f;  // variable assignment
```

```
printf("The profit is: $%f\n", profit); // formatted printout
```

```
The profit is: $2150.479980
```

- By default, `%f` displays the result with six digits. To change it to `p` digits, put `.p` between `%` and `f`. E.g. to print it with 2 digits, `p=2`:

```
float profit;       // type declaration
profit = 2150.48f;  // variable assignment

printf("The profit is: $%.2f\n", profit); // formatted printout
```

```
The profit is: $2150.48
```

- What happens when you get the formatting wrong? in 1, we print a `float` first correctly, then with the wrong format identifier, and then the other way around.

```
float foo;  // defined float
foo = 3.14f;   // assigned float
printf("%.2f\n",foo);  // formatted float as float
printf("%d\n",foo);  // formatted float as int

int bar;  // defined int
bar = 314;   // assigned int
printf("%d\n",bar);  // formatted int as int
printf("%.2f\n",bar);  // formatted int as float
```

```
3.14
   0
 314
 0.0
```

Putting it all together (C program)

- Shipping fees are based on volume instead of weight. For the conversion, the volume is divided by 166. If the result exceeds the actual weight, the shipping fee is based on the "dimensional weight"[5].

- We write a program to compute the dimensional `weight` of a box of given `volume` - we use `/` for division. Let's say the box is 12" x 10" x 8 ". What does 1 need to run?

```
volume = 12 * 10 * 8
  weight = volume / 166
```

- Fixed the errors in 1. The compiler no longer complains, but we don't see anything. How can we print the result?

```
int weight, volume;
volume = 12 * 10 * 8;
weight = volume / 166;
```

- The code in 1 prints the result of the computation.

```
int weight, volume;     // declare variable types
volume = 12 * 10 * 8;   // compute value
weight = volume / 166;  // assign and compute values
printf("The dimensional weight is %d\n",weight); // print result
```

```
The dimensional weight is 5
```

- This is not what we need. When dividing one integer by another, C "truncates" the answer - the result is rounded down, but the shipping company wants us to round up. This can be achieved by adding 165 to the volume before dividing by 166[6] as shown in 1.

```
int weight, volume;     // declare variable types
volume = 12 * 10 * 8;   // compute value
weight = (volume + 165) / 166;  // assign and compute values
printf("The dimensional weight is %d\n",weight); // print result
```

```
The dimensional weight is 6
```

- Now for the final program 1. This time, we allow for tangling the program as `dweight.c`.

```
#include <stdio.h>

int main(void)
{
  // declare variable types
  int height, length, width, volume, weight;

  // variable assignments
  height = 8;
  length = 12;
  width = 10;
  volume = height * length * width;
  weight = (volume + 165) / 166;

  // print results
  printf("Dimensions: %dx%dx%d\n", length, width, height);
  printf("Volume (cubic inches): %d\n", volume);
  printf("Dimensional weight (pounds): %d\n", weight);

  return 0;
}
```

Dimensions: 12x10x8 Volume (cubic inches): 960 Dimensional weight (pounds): 6

Glossary

TERM	EXPLANATION
Variable	Placeholder for a value, e.g. a number
Type	Tells the computer to reserve memory, e.g. `int` for integer numbers
Type declaration	Combination of type and variable name - e.g. `int height;`
`int`	C type for integer numbers, e.g. 2
`float`	C type for floating point numbers, e.g. 3.14
`char`	C type for characters, like "joey"
Formatting	Tells the computer how to print, e.g. `%d` for `int` types
`%d`	Format for integers
`%f` and `%.pf`	Format for floating point numbers (with `p` digits after the point)

References

- N.a. (2020). What is dimensional weight of cargo and why is it important? [blog]. URL: www.unirelo.com.

Footnotes:

[1] In the C99 standard, declarations don't have to come before statements.

[2] The value `8` is called a `constant` because it cannot change

[3] Assignment is variable use. Variable types must be declared before they can be used.

[4] The declaration must precede the use of the variable.

[5]

> "Cargo space has physical limits based on the volume of the cargo and the weight. The reason why both volume & weight are evaluated can be better understood if you consider the cost of shipping a large object with less weight.
>
> For example, a large box containing styrofoam cups weighs very less, i.e., the dimensional (volume) weight of that box will likely be more than its actual weight. It is for this reason that most airlines and other transport providers evaluate both dimensional weight & actual weight, and then use the greater of the two weights to bill you for the transportation costs. The greater of the two weights is also commonly referred to as 'chargeable weight'." (UniRelo, 2020)

[6] 165/166 is 0.9939759, so we've just messed with the actual volume.

Author: Marcus Birkenkrahe
Created: 2022-02-09 Wed 10:46
Validate