CS50 Operators

Overview

You're probably familiar with **operators** from math: the + symbol means addition, the - symbol means subtraction, etc. C also has operators, which you can use to modify or combine values. In addition to having operators that perform basic mathematical operations like addition, subtraction, multiplication, and division, C also has operators that perform other functions: like finding the remainder when dividing, or updating the value of a variable.

Key Terms

- operator
- arithmetic operators
- assignment operators

```
1 | int a = 2 + 8;
2 | int b = 10 - 3;
3 | int c = 4 * 7;
4 | int d = 10 / 2;
5 | int e = 10 / 3;
6 | int f = 13 % 3;
```

a
10

b
7

c
28

d
5

e
3

f
1

Arithmetic Operators

C's **arithmetic operators** perform mathematical functions on numbers. The `+` operator adds two numbers, the `-` operator subtracts one number from another, the `*` operator multiplies two numbers, and the `/` symbol divides one number by another. See lines 1 through 4 of the code to the left to see how such operators work.

When working with `int`s and dividing, it's especially important to be aware that an `int` cannot store non-integer values. For instance, in line 5, we try to store the value of `10 / 3`. C sees a division of two integers, and tries to make the result an integer as well. But since the "real" value of `10 / 3` isn't a whole number, everything after the decimal gets cut off (or "truncated") and `e` is set to just `3`. In order to save the value with the decimal included, we would need to use floating-point numbers, like `float e = 10.0 / 3.0`.

C has another operator, `%`, which is called the modulus operator. The modulus operator gives us the remainder when the number on the left of the operator is divided by the number on the right. Line 6 demonstrates the modulus operator: the remainder when dividing `13` by `3` is `1`, so the value of `f` is set to `1`.

Assignment Operators

C also provides **assignment operators**, which provide a variety of ways to update the value of a variable. The standard assignment operator (`=`) is demonstrated on line 7: it sets the value of `e` to be equal to whatever's on the right side of the equals sign: in this case, the current value of `f` added to `1`.

The variable you're assigning can also be on the right of the equals sign itself. On line 8, the value of `e` is set to the existing value of `e` plus one. While `e = e + 1` might not make logical sense in algebra, it's valid in C. Updating the value of a variable based on its existing value is so common that C has special syntax for it: the operators `+=`, `-=`, `*=`, and `/=` will set a variable to its existing value plus, minus, multiplied by, or divided by some other number.

C also includes special syntax for increasing the value of a variable by one or decreasing the value of a variable by one, by writing the name of the variable followed by `++` or `--`. So a statement like `e++` on line 11 takes the value of `e` and increases it by 1.

```
7  | e = f + 1;
8  | e = e + 1;
9  | e += 1;
10 | e *= 7;
11 | e++;
```

e
2

e
3

e
4

e
28

e
29