Assignment Operators in C

CMSC 104 Section 02
March 27, 2024

Administrative Notes

Quiz 3 next Monday - April 3, 2024

- Covers everything up through today’s lecture

Sample Quiz to be released tomorrow on Blackboard

- Answer key will be posted on GitHub

Homework 5 released today - due next Wednesday (4/3)

Assignment Operators

Here's a short program. Today's lecture is going to be all about what it does, and why

```
#include <stdio.h>
int main () {

  int c = 0;
  printf (" first %5d \n", c++);
  printf("Second %5d \n", ++c);

  return 0;

}
```

When I run this, I get:

```
[arsenaul@linux1 ~/cs104]$ ./a.out
 first     0
Second     2
```

This is all about “syntactic sugar”

Those are little syntax tweaks and tricks we can use to make coding just a little easier.

Here’s the idea: up until now we have done arithmetic using statements like this:

```
i = i+1;
j = j * 5;
k = k / 4;
l = l - 3;
```

That kind of statement happens a lot in C programming, and it’s about 5 keystrokes per command. Surely we can do better.

+=, -=, /=, %= and *=

To cut down on the number of keystrokes, the C language authors created "shortcuts"

`c += 5` is a shortcut meaning `c = c + 5`

`c -= 5` is a shortcut meaning `c = c - 5`

`c *= 5` is a shortcut meaning `c = c * 5`

`c /= 5` is a shortcut meaning `c = c / 5`

`c %= 5` is a shortcut meaning `c = c%5`

Be careful:

+ and * are commutative, so it doesn't matter whether you write "c + 5" or "5 + c"

-, / and % are NOT commutative. Order matters. c/=5 is the same as c = c/5. It is NOT the same as c = 5/c.

5 is just used as an illustration; this can be any expression that evaluates to a number

Let’s keep going - fewer keystrokes!!

In general, using the assignment operator will always let us use fewer keystrokes than writing out the expression:

`c += 5` requires fewer keystrokes than `c = c + 5`

But we can do better. If you spend much time programming, you learn that adding one to a number, and subtracting one from a number, happen ***a lot***.

- E.g, any variable used as a counter gets incremented or decremented

So let’s create special operators just to add one and subtract one

++ and –: the increment and decrement operators

Increment operator: put ++ before or after a variable name, and it gets incremented by one

Decrement operator: put – before or after a variable name, and it gets decremented by one

`c++` is the same as `c = c +1` or `c += 1`

`d--` is the same as `d = d -1` or `d -= 1`

Fewer keystrokes; that's a win!!

Order matters: before or after

Wait - the last slide said you could put the ++ before or after the variable name; what's the difference?

The difference is when you do the incrementing or decrementing:

++ before the variable name means add 1 to the value BEFORE using the value

++ after the variable name means use the value of the variable, and THEN add 1 AFTER you're done.

So that program:

```
#include <stdio.h>
int main () {

  int c = 0;
  printf (" first %5d \n", c++);
  printf("Second %5d \n", ++c);

  return 0;

}
```

++ after the c: use the value of c, which is 0 from the line above. Print out that value - which is why first gets the value 0. Then add 1, which makes the value of c be 1.

++ before the c: the value of c is 1 after the line of code above. Add 1 to that value BEFORE you print out that value. Which is why Second gets the value of 2.

When I run this, I get:

```
[arsenaul@linux1 ~/cs104]$ ./a.out
 first     0
Second     2
```

Now, some practice: make sure you understand

Given: integer variables i = 1; j = 2; k = 3; m = 4

What is the value of

```
i += j + k
```

Now what is the value of

```
j *= k = m + 5
```

And lastly, what is the value of

```
k -= m /= j * 2
```

Hint: when there are multiple equals signs in the expression, you have to work your way from right to left.

Update our Monday example

Take the prime number generator from Monday and update it to use increment and decrement operators