Assignment Operators

Topics
- Increment and Decrement Operators
- Assignment Operators
- Debugging Tips

Reading
- Sections 3.11 - 3.12

Increment and Decrement Operators
- The increment operator ++
- The decrement operator --
- Precedence: lower than (), but higher than * / and %
- Associativity: right to left
- Increment and decrement operators can only be applied to variables, not to constants or expressions

Increment Operator
- If we want to add one to a variable, we can say:
  count = count + 1 ;
- Programs often contain statements that increment variables, so to save on typing, C provides these shortcuts:
  count++ ; OR ++count ;
- Both do the same thing. They change the value of count by adding one to it.
Postincrement Operator

- The position of the ++ determines when the value is incremented. If the ++ is after the variable, then the incrementing is done last (postincrementation).
  
  ```c
  int amount, count;
  count = 3;
  amount = 2 * count++;
  ```

- amount gets the value of $2 \times 3$, which is 6, and then 1 gets added to count.
- So, after executing the last line, amount is 6 and count is 4.

Preincrement Operator

- If the ++ is before the variable, then the incrementing is done first (preincrementation).
  
  ```c
  int amount, count;
  count = 3;
  amount = 2 * ++count;
  ```

- 1 gets added to count first, then amount gets the value of $2 \times 4$, which is 8.
- So, after executing the last line, amount is 8 and count is 4.

Code Example Using ++

```c
#include <stdio.h>

int main ()
{
  int i = 1;
  /* count from 1 to 10 */
  while (i <= 10) {
    printf("%d", i);
    i++;
  }      /* same as ++i */
  return 0;
}
```
If we want to subtract one from a variable, we can say:
\[
\text{count} = \text{count} - 1;
\]
Programs often contain statements that decrement variables, so to save on typing, C provides these shortcuts:
\[
\text{count--} ; \quad \text{OR} \quad --\text{count} ;
\]
Both do the same thing. They change the value of count by subtracting one from it.

Decrement Operator

The position of the -- determines when the value is decremented. If the -- is after the variable, then the decrementing is done last (postdecrementation).

\[
\begin{align*}
\text{int} & \quad \text{amount, count} ; \\
\text{count} & \quad = 3 ; \\
\text{amount} & \quad = 2 \times \text{count--} ;
\end{align*}
\]

amount gets the value of 2 * 3, which is 6, and then 1 gets subtracted from count.
So, after executing the last line, amount is 6 and count is 2.

Postdecrement Operator

If the -- is before the variable, then the decrementing is done first (predecrementation).

\[
\begin{align*}
\text{int} & \quad \text{amount, count} ; \\
\text{count} & \quad = 3 ; \\
\text{amount} & \quad = 2 \times --\text{count} ;
\end{align*}
\]

1 gets subtracted from count first, then amount gets the value of 2 * 2, which is 4.
So, after executing the last line, amount is 4 and count is 2.

Predecrement Operator
A Hand Trace Example

<table>
<thead>
<tr>
<th>Code</th>
<th>Value</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>int answer, garbage = 4; value = value + 1; ++value; answer = 2 * value++; answer = ++value / 2; value--; --value; answer = --value * 2; answer = value-- / 3;</td>
<td>4</td>
<td>garbage</td>
</tr>
</tbody>
</table>

Practice

Given
```
int a = 1, b = 2, c = 3;
```
What is the value of this expression?
```
++a * b - c--
```

What are the new values of a, b, and c?

More Practice

Given
```
int a = 1, b = 2, c = 3, d = 4;
```
What is the value of this expression?
```
++b / c + a * d++
```

What are the new values of a, b, c, and d?
Assignment Operators

<table>
<thead>
<tr>
<th>Statement</th>
<th>Equivalent Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>a = a + 2 ;</td>
<td>a += 2 ;</td>
</tr>
<tr>
<td>a = a - 3 ;</td>
<td>a -= 3 ;</td>
</tr>
<tr>
<td>a = a * 2 ;</td>
<td>a *= 2 ;</td>
</tr>
<tr>
<td>a = a / 4 ;</td>
<td>a /= 4 ;</td>
</tr>
<tr>
<td>a = a % 2 ;</td>
<td>a %= 2 ;</td>
</tr>
<tr>
<td>b = b + ( c + 2 ) ;</td>
<td>b += c + 2 ;</td>
</tr>
<tr>
<td>d = d * ( e - 5 ) ;</td>
<td>d *= e - 5 ;</td>
</tr>
</tbody>
</table>

Practice with Assignment Operators

```c
int i = 1, j = 2, k = 3, m = 4 ;

Expression       Value
i += j + k
j *= k = m + 5
k -= m /= j * 2
```

Code Example Using /= and ++ Counting the Digits in an Integer

```c
#include <stdio.h>
int main ( )
{
    int num, temp, digits = 0 ;
    temp = num = 4327 ;
    while ( temp > 0 )
    {
        printf ("%d
", temp);
        temp /= 10 ;
        digits++ ;
    }
    printf("There are %d digits in %d\n", digits, num);
    return 0 ;
}
```
Debugging Tips

- Trace your code by hand (a **hand trace**), keeping track of the value of each variable.
- Insert temporary `printf()` statements so you can see what your program is doing.
  - Confirm that the correct value(s) has been read in.
  - Check the results of arithmetic computations immediately after they are performed.