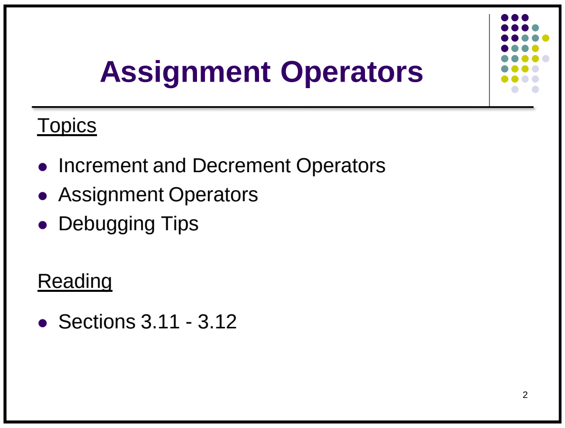
Assignment Operators

CMSC 104, Fall 2012

John Y. Park

1



Assignment Operators

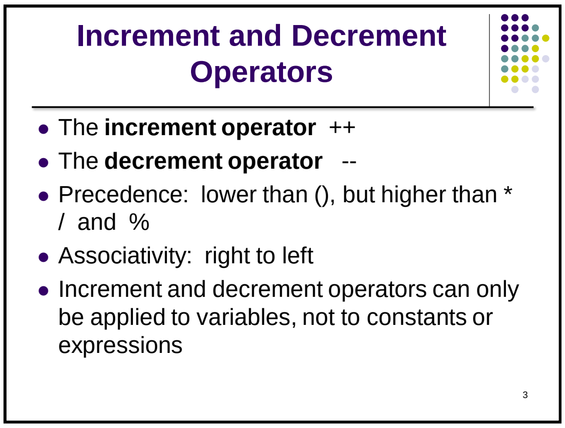
Topics

- Increment and Decrement Operators
- Assignment Operators
- Debugging Tips

Reading

- Sections 3.11 - 3.12

2



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

3

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 (a **postincrement**).

```
int amount, count ;
count = 3 ;
amount = 2 * count++ ;
```

- amount gets the value of 2 * 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 (a **preincrement**).

```
int amount, count ;
count = 3 ;
amount = 2 * ++count ;
```

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

Code Example Using ++

```
#include <stdio.h>
int main ( )
{
    int i = 1 ;

    /* count from 1 to 10 */
    while ( i < 11 )
    {
        printf (“%d  ”, i) ;
        i++ ;                    /* same as ++i */
    }
    return 0 ;
}
```

Decrement Operator

- If we want to subtract one from a variable, we can say:

  count = count - 1 ;

- Programs often contain statements that decrement 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 subtracting one from it.

Postdecrement Operator

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

```
int amount, count ;
count = 3 ;
amount = 2 * count-- ;
```

- 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.

Predecrement Operator

- If the -- is before the variable, then the decrementing is done first (a **predecrement**).

```
int amount, count ;
count = 3 ;
amount = 2 * --count ;
```

- 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.

A Hand Trace Example

int answer, value = 4 ;

Code	Value	Answer
	4	garbage
value = value + 1 ;		
value++ ;		
++value ;		
answer = 2 * value++ ;		
answer = ++value / 2 ;		
value-- ;		
--value ;		
answer = --value * 2 ;		
answer = value-- / 3 ;		

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?

13

Assignment Operators

= += -= *= /= %=

Statement	Equivalent Statement
a = a + 2 ;	a += 2 ;
a = a - 3 ;	a -= 3 ;
a = a * 2 ;	a *= 2 ;
a = a / 4 ;	a /= 4 ;
a = a % 2 ;	a %= 2 ;
b = b + (c + 2) ;	b += c + 2 ;
d = d * (e - 5) ;	d *= e - 5 ;

14

Practice with Assignment Operators

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

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

15

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

```
#include <stdio.h>
int main ( )
{
    int num, temp, digits = 0 ;
    temp = num = 4327 ;
   while ( temp > 0  )
    {
        printf (“%d\n”, 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.