Assignment and Arithmetic Operators
What are operators?

- Operators allow us to modify and manipulate information.
- They are symbols that represent a commonly used operation, such as addition
  - $2 + 2$
Assignment Operator

- "=" is used to assign a value to a variable

```c
int bob = 5;
tax = income * RATE;
RATE = .05;  // NO; RATE is const
4 = income + 64;  // NO, can't modify 4
```
Steps of Execution

tax = income * RATE;
Steps of Execution

tax = income * RATE;
Steps of Execution

\[ \text{tax} = \text{income} \times \text{RATE}; \]
Steps of Execution

\[
\text{tax} = \text{income} \times \text{RATE};
\]
Casting

int someValue;
double Num1, Num2;
someValue = Num1 + Num2;
someValue = static_cast<int>(Num1 + Num2);
Arithmetic Operators

- "+", "-", "*", "/", "%" (modular arithmetic)

someValue = num1 + num2;  // addition

someValue = num1 - num2;  // subtraction

someValue = num1 * num2;  // multiplication

someValue = num1 / num2;  // division

someValue = num1 % num2;  // modulus
Type Promotion

```c
int integer1;
float float1;

float1 + integer1 // gives a float
float1 - integer1 // gives a float
float1 * integer1 // gives a float
float1 / integer1 // gives a float
integer1 / float1 // gives a float
float1 % integer1 // can’t be done
integer1 % float1 // can’t be done
```
Example 1

float celc;
int fahr;

celc = (5/9)*(fahr-32);
celc = (5.0/9)*(fahr-32);
float average_age;
int total_of_ages, num_people;

average_age = total_of_ages / num_people;
average_age = static_cast<float>(total_of_ages) / num_people;
Modular Arithmetic

- If \( a \% b \), then the result is the remainder of \( a / b \)
  - \( 4\%7 \) is 4 (since \( 4/7 \) is 0 with remainder 4)
  - \( 7\%3 \) is 1 (since \( 7/3 \) is 2 with remainder 1)
  - \( 27\%3 \) is 0 (since \( 27/3 \) is 9 with remainder 0)
- Cool trick...

```c
int tens_digit;
tens_digit = (x\%100)/10;
```
Increment and Decrement

- \( \text{val} = \text{val} + 1; \)
  - \( \text{val}++; \) or \( ++\text{val}; \)

- \( \text{val} = \text{val} - 1; \)
  - \( \text{val}--; \) or \( --\text{val}; \)
Pre- vs. Post-

```c
int val = 6;
int num;
num = ++val;

int val = 6;
int num;
num = val++;
```
Other Fast Operators

x += y;  // equivalent to x = x + y;
x -= y;  // equivalent to x = x - y;
x /= y;  // equivalent to x = x / y;
x *= y;  // equivalent to x = x * y;
x %= y;  // equivalent to x = x % y;
End of Session