Arithmetic Operators

Section 2.15 & 3.2
p 60-63, 81-89
Today

• Arithmetic Operators & Expressions
  ○ Computation
  ○ Precedence
  ○ Algebra vs C++
  ○ Exponents
Assigning float to int

```cpp
int intVariable;

intVariable = 42.7;

cout << intVariable;
```

• What do you think is the output?
Assigning `double` to `int`

- What is the output here?

```cpp
int intVariable;

double doubleVariable = 78.9;
intVariable = doubleVariable;

cout << intVariable;
```
Integer Division

• What is the output?

  ○ `int grade;
    grade = 100 / 20;
    cout << grade;

  ○ `int grade;
    grade = 100 / 30;
    cout << grade;`
Division

• $\text{grade} = 100 \ / \ 40$;
  - Check operands of /
    - the data type of grade is not considered, why?
  - We say the integer is *truncated*.

• $\text{grade} = 100.0 \ / \ 40$;
  - What data type should grade be declared as?
Mathematical Expressions

- Complex mathematical expressions are created by using multiple operators and grouping symbols
  - expression: programming statement that has a value
    - `sum = 21 + 3;`
    - `number = 3;`

In these two examples, we assign the value of an expression to a variable.
Arithmetic Operators

- Operators allow us to manipulate data
  - Unary: \textit{operator operand}
  - Binary: \textit{operand operator operand (left hand side)} \textit{operand (right hand side)}

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Negation</td>
<td>Unary</td>
<td>- 5</td>
</tr>
<tr>
<td>=</td>
<td>Assignment</td>
<td>Binary</td>
<td>rate = 0.05</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>Binary</td>
<td>cost * rate</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>Binary</td>
<td>cost / 2</td>
</tr>
<tr>
<td>%</td>
<td>Modulus</td>
<td>Binary</td>
<td>cost % 2</td>
</tr>
<tr>
<td>+</td>
<td>Addition</td>
<td>Binary</td>
<td>cost + tax</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>Binary</td>
<td>total - tax</td>
</tr>
</tbody>
</table>
Operator Precedence

• \( \text{result} = 4 \times 2 - 3; \)

• \( \text{result} = 12 + 6 / 3; \)
  ○ \( \text{result} = \) ?

• Rules on how to evaluate an arithmetic expression
  ○ arithmetic expressions are evaluated left to right
  ○ do them in order of precedence
  ○ grouping symbols ( )
### Operator Precedence

#### Precedence of Arithmetic Operators
(Highest to Lowest)

<table>
<thead>
<tr>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>(unary negation) -</td>
</tr>
<tr>
<td>*       /       %</td>
</tr>
<tr>
<td>+       -</td>
</tr>
<tr>
<td>(assignment) =</td>
</tr>
</tbody>
</table>

If two operators have the same precedence, evaluate them from left to right as they appear in the expression.
Practice

```cpp
int x = 3;
double y = 2.5;

cout << 5 + 2 * 3;
cout << (10 / 2 - y);
cout << 3 + 12 * 2 - 3;
cout << 4 + 17 / 3.0 + 9;
cout << (6 - y) * 9 / x * 4 - 9;
```

If you are unsure, you can always type up and run the code in Visual Studio.
Modulus

• Modulus is the remainder after integer division

• grade = 100 % 20;
  ○ grade = ?

• grade = 100 % 30;
  ○ grade = ?

• rem = x % n;
  ○ What are the possible values for rem?
Summary

• Today we have looked at:
  ○ Arithmetic Operators & Expressions

• Next time we will:
  ○ Continue looking at mathematic operators

• Completed section 2.15 & started on section 3.2