Today

- Last time we learned about two types of executable statements
  - Assignment statements
  - Input/output statements
- Today we will
  - Cover arithmetic statements

Arithmetic Expressions

- Arithmetic expressions manipulate numeric data
- We’ve already seen simple ones
- The main arithmetic operators are
  - Addition
  - Subtraction
  - Multiplication
  - Division
  - Modulus

+,-, and *

- Addition, subtraction, and multiplication behave in C++ in the same way that they behave in algebra

```cpp
int num1, num2, num3, num4, sum, mul;
num1 = 3;
um2 = 5;
um3 = 2;
um4 = 6;
sum = num1 + num2;
mul = num3 * num4;
```

Division /

- The division operator can be used with both integers and doubles
- If the operands are both **doubles**, the result is a **double**
  - Example: 7.0 / 2.0 is 3.5
- If the operands are both **ints**, the result is an **int**
  - Any fractional part in integer division is discarded
  - Example: 7 / 2 is 3
- If mixed, the **int** operand is converted to a **double** and the result is a **double**
  - Example: 5 / 2.5 is 2.0

Division

- Divisor (second operand) cannot be 0
- Division with negative integers may or may not be allowed
  - It depends on the compiler
Modulus
- % returns the integer remainder of integer division
- Both operands must be integers
- If second operand is negative, results will vary from system to system

Examples
- $3 \div 5 = 5 \mod 3 = 0$
- $4 \div 5 = 5 \mod 4 = 1$
- $5 \div 5 = 15 \mod 5 = 0$
- $6 \div 5 = 15 \mod 6 = 3$
- $7 \div 5 = 8 \mod 0 = 0$
- $15 \div -7 = -2$

Arithmetic Operations
- Arithmetic operations in C++ must be entered in straight-line form
- Algebraic notation is not accepted by the compiler
  - Instead, you should use: $x = 3 / 4 * 2$;
  - If we wanted to evaluate $x = \frac{3}{4*2}$, then we would use $x = 3 / (4 * 2)$;

Expressions with Multiple Operators
- Example: $x = 5 + 3 * 2 - 1$;
- What’s the value of $x$?
- There are rules for the order of evaluation so every computer will calculate the same expression the same way every time

Unary Operators
- Most of the examples that we have seen so far use binary operators
  - i.e. they take two operands, one on the left of the operator and one on the right of the operator
- C++ also supports unary operators
  - i.e. operators that take one operand to the right of the operator
- Positive and negative are examples of unary operators
  - $x = -5$;
  - $y = -x$;

Operator Precedence
- Anything in parentheses is evaluated first
  - Innermost first
- Operator precedence
  - Unary operators +, -
  - Binary operators *, /, %
  - Binary operators +, -
- Why is operator precedence important?

Operator Associativity
- Operator associativity refers to the order in which expressions of the same level are evaluated
- Binary operators are evaluated left to right
- Unary operators are evaluated right to left
- Give a numerical example that illustrates the importance of operator associativity
Example

- In what order is the following expression evaluated?
  - $num = x \times y \times z + a / b - c \times d$
- What is the value of $num$ if
  - $x = 2, y = 3, z = 2, a = 4, b = 2, c = 5, d = 2$

Problem

- How would you write the expression $y = x^3 + 7$ in C++ where ^ stands for exponentiation?
- What is the value of $x$ in the following expressions, assume $x$ is a double
  - $x = 5.0 * 3.0 / 2.0;$
  - $x = 3.0 + 2.0 - 5.0;$

string Data Type

- Before we move on, I want to introduce you to a useful data type
- This is not a primitive data type, but you can use it by including a C++ library as a preprocessor directive
  - The library you need to include is
    ```
    #include <string>
    ```
- The string data type is used to create variables that can hold multiple characters
- String name = “george”;

string Data Type

```
#include "stdafx.h"
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string name;
    cout << "what is your name?";
    cin >> name;
    cout << "Hello " << name << endl;
    return 0;
}
```

Problem

- Write a C++ program that allows the user the ability to enter their name and the number of nickels and pennies they have. You are then to print the number of dollars and change that corresponds to.

Summary

- In today’s lecture we learned
  - How to write complex arithmetic expressions
  - About operator precedence and operator associativity
  - About the modulus operator
  - About string variables
- We have covered p. 31 - 34 of your textbook