Arithmetic operators in C++

Objectives of the Lecture
- Arithmetic operators and Operator Precedence.
- Assignment statement.
- Increment and decrement operators.
- Syntax Errors in C++ program.
- Documentation in C++ program.
- Programming Example: Convert Length

Arithmetic Operators and Operator Precedence

- C++ arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - % modulus operator
- +, -, *, and / can be used with integral and floating-point data types.
- % can be used only with integral data types.
- Operators can be unary or binary.
- Order of Precedence
  - All operations inside of () are evaluated first
  - *, /, and % are at the same level of precedence and are evaluated next
  - + and – have the same level of precedence and are evaluated last
  - When operators are on the same level: performed from left to right (associativity)
    - 3 * 7 – 6 + 2 * 5 / 4 + 6 means
      - (((3 * 7) – 6) + ((2 * 5) / 4)) + 6
- Expressions
  - If all operands are integers, expression is called an integral expression and yields an integral result, for example: 2 + 3 * 5
  - If all operands are floating-point, expression is called a floating-point expression and yields a floating-point result, for example: 12.8 * 17.5 - 34.50
  - If the expression has operands of different data types (integers and floating-point), expression is called mixed expression, examples of mixed expressions are:
    - 2 + 3.5
    - 6 / 4 + 3.9
    - 5.4 * 2 - 13.6 + 18 / 2
Assignment Statement

- The assignment statement takes the form:

\[ \text{variable} = \text{expression}; \]

- Expression is evaluated and its value is assigned to the variable on the left side.
- In C++, \( = \) is called the assignment operator.

**Example 2-13**

```cpp
int num1, num2;
double sale;
char first;
string str;

num1 = 4;
num2 = 4 * 5 - 11;
sale = 0.02 * 1000;
first = 'D';
str = "It is a sunny day.";
```

**Example 2-14**

1. num1 = 18;
2. num1 = num1 + 27;
3. num2 = num1;
4. num3 = num2 / 5;
5. num3 = num3 / 4;

- C++ has special assignment statements called **compound assignments**  
  \( +=, -=, *=, /=, \text{ and } %= \)

Example:

```cpp
x *= y;
```

Increment and Decrement Operators

- **Increment operator**: increment variable by 1
  - **Pre-increment**: ++variable
  - **Post-increment**: variable++

- **Decrement operator**: decrement variable by 1
  - **Pre-decrement**: --variable
  - **Post-decrement**: variable--

What is the difference between the following?

```cpp
x = 5;
y = ++x;
```

and

```cpp
x = 5;
y = x++;
```
Syntax Errors in C++ program

Errors in syntax are found in compilation

```cpp
int x;  //Line 1 OK
int y  //Line 2: error
double z;  //Line 3 OK
y = w + x;  //Line 4: error
```

Documentation in C++ program

A well-documented program is easier to understand and modify
You use comments to document programs
Example:

```cpp
int feet;  //variable to hold given feet
int inches;  //variable to hold given inches
int totalInches;  //variable to hold total inches
double centimeters;  //variable to hold length in
```

Programming Example: Convert Length

Write a program that takes as input a given length expressed in feet and inches and convert and outputs the length in centimeters

Problem analysis:
- **Input**: length in feet and inches
  - Lengths are given in feet and inches.
  - Convert the length in feet and inches to all inches:
    - Multiply the number of feet by 12
    - Add given inches
  - One inch is equal to 2.54 centimeters
- **Output**: equivalent length in centimeters
  - Program computes the equivalent length in centimeters

- **Needed variables**
  - `int feet;  //variable to hold given feet`
  - `int inches;  //variable to hold given inches`
  - `int totalInches;  //variable to hold total inches`
  - `double centimeters;  //variable to hold length in centimeters`

- **Named Constant**
  - `const double CENTIMETERS_PER_INCH = 2.54;`
  - `const int INCHES_PER_FOOT = 12;`

- **Programming Example: Body of the Function**
  - The body of the function main has the following form:
    ```cpp
    int main ()
    {
      declare variables
      statements
    }
    ```
return 0;
}

using namespace std;

// Named constants
const double CENTIMETERS_PER_INCH = 2.54;
const int INCHES_PER_FOOT = 12;

int main ()
{
    // Declare variables
    int feet, inches;
    int totalInches;
    double centimeter;

    // Statements: Step 1 - Step 7
    cout << "Enter two integers, one for feet and "
         << "one for inches: ";          // Step 1
    cin >> feet >> inches;            // Step 2
    cout << endl;
    cout << "The numbers you entered are " << feet
         << " for feet and " << inches
         << " for inches. " << endl;     // Step 3

    totalInches = INCHES_PER_FOOT * feet + inches; // Step 4

    cout << "The total number of inches = "
         << totalInches << endl;       // Step 5

    centimeter = CENTIMETERS_PER_INCH * totalInches; // Step 6

    cout << "The number of centimeters = "
         << centimeter << endl;        // Step 7

    return 0;
}